Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Création d'un Outil Interactif d'Analyse de Données Patients

Volvert, Sylvain

Award date:
2015

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 05. May. 2023

Université de Namur
Faculté d'informatique
Année académique 2014–2015

Création d'un Outil Interactif d'Analyse de Données Patients

Sylvain VOLVERT

Maître de stage : James Ortiz

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Pierre-Yves Schobbens

Co-promoteur : James Ortiz

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Note

Ce document présente des données fictives. Les prénoms, noms, dates et lieux de naissance, les éventuelles maladies ou toute autre information mentionnée dans ce document concernant des patients ne correspondent EN AUCUN CAS à des personnes réelles. Toute similarité avec quiconque serait purement fortuite.

Résumé

Dans le milieu médical, les traitements et les médicaments que nous connaissons aujourd'hui ont été conçus puis testés lors d'études cliniques. Qu'elles soient à l'origine d'un groupe pharmaceutique ou de l'hôpital lui-même, les études cliniques sont une étape capitale dans l'évolution des traitements. Malheureusement, l'informatique, n'est pas encore suffisamment présente dans cette branche.

Ce mémoire vise fournir une compréhension détaillée de plusieurs concepts clés qui peuvent améliorer les résultats d'une étude clinique à travers la création de programmes informatiques d'un genre nouveau. De tels logiciels pourraient aider les médecins dans leurs décisions. Dans ce sens, un logiciel interactif d'analyse de données patients, pour la recherche clinique, a été conçu.

Mots clés : études cliniques, vMR, ontologie, data mining, web sémantique.

Abstract

In the medical field, current treatments and drugs were developed and tested during clinical studies. Whether they are made by a pharmaceutical group or by a hospital itself, clinical studies are a crucial step in the development of new treatments. Unfortunately, IT is not yet sufficiently present in this domain.

This thesis aims to provide a detailed understanding of several concepts that can improve the results of a clinical study through the creation of a new kind of software. This kind of software could help doctors in their decisions. In that way, an interactive software for analyzing patient data for clinical research was designed.

Keywords: clinical study, vMR, ontology, data mining, semantic web.

Avant-Propos

Ce mémoire est le fruit de quatre mois de recherches et de développement lors du stage que nous avons effectué à la Faculté d'Informatique de l'Université de Namur (Belgique). Ce stage, qui constitue une étape importante dans le cursus universitaire, nous a permis de nous rendre compte du travail de l'informaticien en entreprise. Il a été pour nous une expérience enrichissante tant au niveau professionnel que personnel.

L'ouvrage retrace les activités principales qui se sont déroulées durant le stage avec un accent sur ce qui existe déjà au niveau informatique et un accent sur l'implémentation de notre logiciel. Afin d'éclaircir certains points, une notation a été choisie pour quelques éléments. Ainsi,

— Langage représente un langage de programmation ;
— Logiciel représente un logiciel, une librairie ou une plateforme ;
— Extension représente une extension de fichier.

Nous tenons à remercier notre promoteur, Pierre-Yves Schobbens ainsi que notre maître de stage et co-promoteur, James Ortiz, pour leur aide, leurs conseils et leur implication tout au long du stage et de la rédaction de ce mémoire. Nous tenons également à remercier toutes les personnes qui nous ont aidé à travers les différentes réunions dont Benoît Frénay, Thierry Vander Borght et Laurent Grignet qui nous ont guidé dans la réalisation de ce travail.

Nous voudrions aussi remercier ces quelques personnes pour les connaissances et les conseils qu'ils nous ont apportés : Pol Benats, Nicolas Genon, Fabian Gilson, Amanuel Koshima et Aubry Volvert. Nous pensons tout particulièrement aux doctorants, qui partageaient le même bureau, pour nous avoir supportés durant ce stage mais aussi pour la bonne ambiance qui y régnait. Ensuite, nous voudrions évoquer tous les professeurs et assistants de l'Université de Namur qui ont contribué à notre formation durant le baccalauréat et le master, et grâce à qui le travail effectué n'aurait pas été de même qualité.

Nous tenons à remercier également Françoise, Jacqueline, Amélie, Adélaïde et Marianne pour la relecture de ce document. Enfin, nous pensons tout particulièrement à nos familles et amis pour leur soutien et leur patience durant nos études, et surtout pendant la rédaction de ce mémoire. Un tout grand merci à eux.

Table des matières

Table des figures

Table des codes

Liste des acronymes

ADL	Archetype Definition Language
ANSI	American National Standards Institute
API	Application Programming Interface
ARFF	Attribute-Relation File Format
CDS	Clinical Decision Support
CDSS	Clinical Decision Support System
DAML+OIL	DARPA Agent Markup Language + Ontology Inference Layer
DPI	Dossier Patient Informatisé
EHR	Electronic Health Record
EMR	Electronic Medical Record
HL7	Health Level Seven
IFLEC	Tableau de Bord pour la Recherche Clinique
ISO	International Organization for Standardization
KMEHR	Kind Messages for Electronic Healthcare Record
OIL	Ontology Inference Layer
OLED	OntoUML Lightweight EDitor
OWL	Web Ontolgy Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RIM	Reference Information Model
SPARQL	SPARQL Protocol and RDF Query Language
sumEHR	Summarised Electronic Health Record
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
vMR	Virtual Medical Record
W3C	World Wide Web Consortium
Weka	Waikato Environment for Knowledge Analysis
XML	eXtensible Markup Language
XSD	XML Schema Definition

Glossaire

- **Agrégation :** En informatique, l'agrégation permet de définir une entité comme étant un assemblage de plusieurs sous-entités.

- **Architecture :** Une architecture correspond à la manière dont un système informatique a été conçu. Comme une maison, un système a besoin de fondations solides pour permettre son évolution. Une bonne architecture, pour un logiciel, peut éviter les maintenances longues et difficiles.

- **Data mining :** Le concept de *data mining* tend à extraire de l'information parmi les gigantesques amas de données. Particulièrement en vogue actuellement, le data mining est nécessaire pour fouiller les bases de données qui ne cessent de s'étendre. Des algorithmes particuliers existent pour réaliser cette tâche complexe.

- **Dossier Médical Informatisé (DMI) :** Le DMI est un fichier numérique contenant toutes les informations concernant l'historique médical ainsi que les caractéristiques médicales d'un patient. Ces fichiers sont essentiels pour les établissements de soins.

- **Dossier Patient Informatisé (DPI) :** Synonyme de Dossier Médical Informatisé (DMI).

- **Electronic Health Record (EHR) :** Synonyme de Dossier Médical Informatisé (DMI).

- **Electronic Medical Record (EMR) :** Synonyme de Dossier Médical Informatisé (DMI).

- **Étude clinique :** Une étude clinique est la seule méthode grâce à laquelle les firmes pharmaceutiques et les hôpitaux peuvent en savoir plus sur le traitement d'une maladie. Elle se pratique généralement dans les hôpitaux à l'initiative d'un sponsor qui souhaite par exemple tester l'efficacité d'un médicament. Il existe deux types d'études : les prospectives et les rétrospectives.

- **Exploration de données :** Voir Data mining.

- **Framework :** Un framework ou une structure logicielle est un ensemble cohérent de composants logiciels, qui sert à créer les fondations ainsi que les grandes lignes d'un autre logiciel. Le principal utilisateur d'un framework est le programmeur.

- **Hasher :** Ce terme désigne une fonction d'exécution sur les mots de passe. Des calculs sont réalisés sur un mot de passe reçu en entrée. Le résultat qui en résulte est une suite de caractères (le *hash*) incompréhensible pour l'être humain. L'avantage des algorithmes qui implémentent ces fonctions de *hashage* réside dans le fait qu'il est impossible de réaliser l'opération inverse, c'est-à-dire retrouver le mot de passe d'origine.

- **HL7 :** HL7 ou Health Level 7 est une organisation qui s'occupe de fournir des standards internationaux dans le domaine de la santé pour les échanges de données médicales au format numérique.

- **IFLEC :** Il s'agit d'un projet de la région Wallonne qui a débuté en septembre 2013 pour une période de deux ans. Ce projet fait intervenir l'Université de Namur et l'entreprise MIMS afin de créer un outil interactif d'analyse de données patients à partir d'un dossier médical informatisé.

- **Mapping :** Faire un mapping signifie faire des correspondances entre plusieurs éléments.

- **NoSQL :** NoSQL (Not only SQL) désigne une catégorie de systèmes de gestion de base de données (SGBD) qui ne repose plus sur l'architecture classique des bases relationnelles. L'unité logique n'est plus la table, et les données ne sont en général pas manipulées avec SQL.

- **Ontologie :** Une ontologie est une structuration des concepts d'un domaine bien particulier qui vise à offrir une base solide dans la représentation du domaine en question. Elle permet de décrire formellement la connaissance de tous les concepts utiles au domaine, en créant des relations entre les éléments.

- **Pattern :** Le mot *pattern* est souvent utilisé pour désigner un modèle, une structure, etc. Il désigne tantôt un modèle ou une organisation des données pouvant apparaître dans les graphiques de RetroStudy, tantôt une structuration des éléments dans l'architecture des applications. Pour cette deuxième définition, le mot *design pattern* est plus utilisé.

- **Protocole (d'étude) :** Le protocole d'étude est un document qui décrit le déroulement d'une étude clinique avec les objectifs précis, le déroulement et les méthodes de traitements. Il inclut parfois des informations scientifiques et des statistiques chiffrées.

- **Relation is-a :** Les relations *is-a*, utilisées dans les bases de données, permettent de définir les hiérarchies qui pourraient exister entre plusieurs éléments. Dans la littérature, les termes *sur-type* et *sous-type* font directement référence aux relations *is-a*.

- **Requête :** Dans le cas de RetroStudy, une requête correspond aux contraintes que le médecin désire créer dans le formulaire dynamique. Le nom « requête » vient du fait que les contraintes sont transformées en une requête SQL qui est envoyée directement à la base de données.

- **Standalone :** Se dit d'un logiciel qui est autonome, c'est-à-dire un logiciel qui fonctionne sans qu'un autre soit nécessaire à son fonctionnement.

- **Système de Gestion de Bases de Données (SGBD) :** Un SGBD est un logiciel destiné à stocker et à partager des informations dans une base de données, en garantissant la qualité, la pérennité et la confidentialité des informations, tout en cachant la complexité des opérations. Les SGBD les plus connus sont Oracle, MySQL ou encore PostgreSQL.

- **URI :** Semblable à une URL, une URI (*Uniform Resource Identifier*) permet d'identifier par une courte chaîne de caractères un éléments situés sur le web. Elle doit permettre l'identification d'une ressource de manière permanente, même si la ressource est déplacée ou supprimée. Les URL sont un sous-ensemble des URI.

- **URL :** Le sigle URL (*Uniform Resource Locator*) désigne une chaîne de caractères utilisée pour adresser les ressources du World Wide Web comme par exemple les documents HTML. Les URL constituent un sous-ensemble des identifiants uniformisés de ressource (URI).

- **Workflow :** Signifiant littéralement *Flux de travaux*, le workflow sert à représenter une suite de tâches ou d'opérations effectuées par une personne ou un organisme. Le flux permet de représenter schématiquement les différentes phases les unes à la suite des autres.

Introduction

Avant de plonger directement dans le vif du sujet, des précisions sont à émettre en ce qui concerne tout particulièrement le cadre et le contexte de ce mémoire. Ce chapitre vise donc à définir le cadre spacio-temporel ainsi que les notions qui sont nécessaires pour permettre la bonne compréhension de tous les éléments figurant dans ce document.

Contexte

De nos jours, l'informatique est présente dans quasi tous les domaines et ne cesse de se répandre. Présente dans la maison, l'école, les entreprises, elle peine à se faire une place dans les hôpitaux surtout pour l'aide à la décision. Alors que le médecin reste seul face au diagnostic, l'informatique pourrait, dans certaines circonstances, appuyer le médecin dans ses décisions. Le contexte concerne donc pleinement le domaine médical. Avant de prétendre à la création d'un véritable outil sophistiqué qui serait le remplaçant du médecin, la recherche doit encore continuer. Pour un jour y arriver, un cadre plus restrictif du domaine médical doit être sélectionné : les études cliniques. Grâce à ces études cliniques, la recherche avance dans les traitements des maladies.

« *Les protocoles de recherches cliniques (appelés aussi études cliniques), permettent de déterminer si les nouveaux médicaments ou traitements, ainsi que l'application inédite de traitements déjà connus, sont sûrs, efficaces et porteurs de mieux-être. La recherche clinique est la seule méthode grâce à laquelle on peut en savoir plus sur le traitement de la maladie chez le patient atteint d'une maladie grave comme d'une pathologie plus bénigne.* » [1]

Les études cliniques englobent deux types d'études : les études prospectives et les études rétrospectives. Les études prospectives sont souvent créées à l'initiative d'une firme pharmaceutique ou par l'hôpital dont l'objectif a été défini avant le début de l'étude clinique. Cet objectif est défini dans un document que l'on appelle « Protocole d'étude ». Ce document décrit et définit l'étude dans sa globalité comme par exemple son coût, sa progression, le nombre de patients ou encore l'agencement des examens que le patient doit subir. L'autre type d'étude, celui des études rétrospectives, n'a pas de but clairement défini à priori. En effet, une recherche est faite sur un ensemble de données patients pour les analyser et trouver des cas pouvant correspondre aux critères introduits. Ce stage concerne pleinement les études rétrospectives. Une représentation graphique des deux types d'études est disponible à la figure 1.

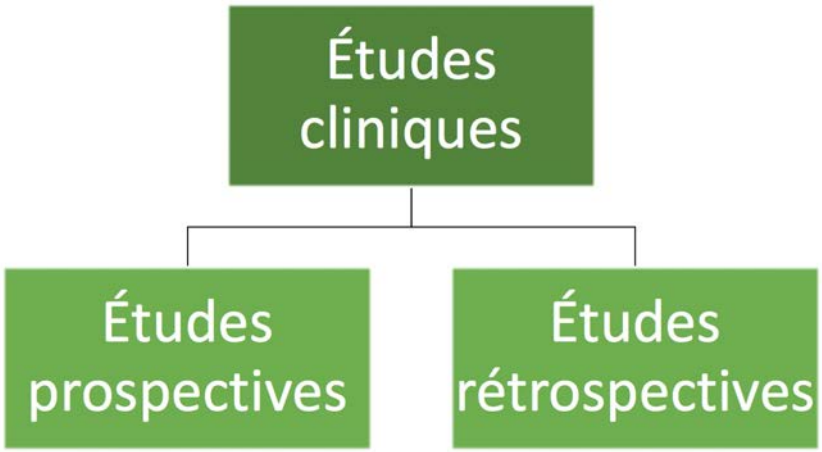

Figure 1 – Les deux types d'études cliniques

Avant de réaliser ce mémoire, nous avions déjà eu l'occasion d'étoffer notre compréhension du domaine médical lors d'un cours donné en troisième année de baccalauréat. Ce cours permettait aux étudiants de réaliser un logiciel suivant les exigences d'un client. Nous avons eu la chance de pouvoir participer, avec les Cliniques Universitaires de Mont-Godinne, à l'élaboration d'un logiciel permettant aux médecins de créer, modifier et parcourir des données qui concernent les études cliniques. Considéré comme le prélude de ce stage et de ce mémoire, le logiciel réalisé à cette époque nous a permis de mieux cerner la problématique des études cliniques.

Un projet de la région Wallonne, nommé IFLEC (*Tableau de Bord pour la Recherche Clinique*), est à la base du mémoire. Ce projet fait intervenir deux partenaires que sont l'Université de Namur et l'entreprise privée MIMS. Cette dernière est une société éditrice de logiciels pour le secteur de la santé, active depuis 1997. Elle conçoit des logiciels qui optimisent la prise en charge des patients à l'hôpital [2]. Pendant 2 ans, l'objectif du projet IFLEC était de développer une maquette permettant l'extraction de données depuis un dossier patient informatisé. Ces données devaient ensuite être injectées dans un serveur d'analyse pour permettre aux responsables des études cliniques d'y effectuer des recherches et d'en extraire les résultats. Le développement d'une telle maquette correspond à la création d'un outil interactif d'analyse de données patients.

Le projet IFLEC a débuté en septembre 2013 et comptait plusieurs phases. L'une des phases était de concevoir un premier logiciel dans le but de faciliter la gestion administrative des études cliniques prospectives et informatiser la définition des protocoles nécessaires au bon déroulement d'une étude. Cette phase a fait l'objet d'un mémoire, réalisé par un étudiant de la Faculté d'informatique à l'Université de Namur en 2014 [3]. Alors que son mémoire et son logiciel concerne les études cliniques prospectives, le cadre du projet IFLEC veut recouvrir les deux types d'études. Pour ce faire, une autre phase du projet consistait à se concentrer sur les études rétrospectives pour fournir un outil d'analyse de données. C'est dans ce cadre que s'inscrit pleinement ce mémoire.

Ce document vise à répondre à un besoin de recherches et d'obtention de résultats précis quant aux requêtes des médecins pour l'éventuelle faisabilité d'une étude clinique (rétrospective). Pour ce faire, un logiciel sur mesure a été conçu. Ce logiciel se nomme RetroStudy et tente de répondre à ce besoin. Le but de ce mémoire est d'essayer de comprendre comment, et de quelle manière, un outil interactif d'analyse de données peut être élaboré mais surtout de présenter un prototype fonctionnel.

Contenu du mémoire

Cet ouvrage est décomposé en deux grandes parties représentant **l'état de l'art**, qui décrit plusieurs technologies et tente de familiariser le lecteur avec celles-ci et **l'implémentation**, qui détaille le processus de création du logiciel.

État de l'art Dans le **chapitre 1** seront expliqués la problématique de la gestion et du stockage des données dans les hôpitaux ainsi que les différents standards qui existent à l'heure actuelle. Dans le **chapitre 2**, il sera question des ontologies qui comportent de nombreux avantages dans leur utilisation. Enfin dans le **chapitre 3**, la manière dont les données peuvent être traitées et explorées sera expliquée en détail sous le concept de *data mining*.

Implémentation La deuxième partie du mémoire traite les informations présentées dans la première partie au niveau plus pratique et détaille la conception du logiciel RetroStudy. Dans le **chapitre 4**, l'architecture de l'application sera exposée. Dans le **chapitre 5**, le choix d'un standard pour stocker les données au sein des hôpitaux sera choisi et les opérations d'implémentation pour arriver à une base de données fonctionnelle sera également proposée. Dans le **chapitre 6**, le sujet principal sera le *data mining* au niveau pratique et donc au niveau des algorithmes qui ont été implémentés dans RetroStudy. Pour terminer cette deuxième partie, le **chapitre 7** présentera surtout des illustrations de l'application à travers un parcours typique par l'utilisateur.

En guise de conclusion, le **chapitre 8** résumera les notions vues dans ce document et proposera en outre les améliorations possibles du logiciel. La continuité de la recherche et des travaux en lien avec ce mémoire y fera également mention.

Première partie

État de l'Art

Chapitre 1

Stockage des données

Ce premier chapitre met en avant la problématique du stockage des données au sein des hôpitaux mais aussi celle de la transmission et l'échange de ces données entre différentes institutions. En premier lieu, une section éclaircira la problématique du stockage des données dans le milieu hospitalier. Ensuite, une autre fera la synthèse des standards qui existent actuellement dans le monde. Enfin, la dernière section sélectionnera le meilleur standard et présentera les avantages et inconvénients de chaque élément.

1.1 Les Données Médicales

Le domaine médical est un domaine complexe faisant interagir de nombreux composants logiciels entre eux, où la manière de stocker les données est souvent fort différente. Il est clair que le cheminement de l'information à travers les différents systèmes des hôpitaux est complexe. Aussi, la nécessité d'un dossier médical informatisé donne lieu à réorganiser la façon dont l'information est enregistrée localement et transmise à d'autres institutions. Les deux sous-sections suivantes expliciteront cela davantage.

1.1.1 Cheminement de l'Information

Le projet IFLEC a analysé la complexité des infrastructures au sein de l'hôpital universitaire de Mont-Godinne. Il en ressort une vision claire de l'architecture logicielle et matérielle. En guise d'illustration, la figure 1.1 présente le cheminement

complet de l'information (*workflow*) depuis l'arrivée d'un patient jusqu'à la mise à disposition des dossiers patients sur le réseau de la clinique. Sur ce schéma, les unités logicielles de traitement de l'information, comme la planification des examens ou le stockage des images, sont représentées [4].

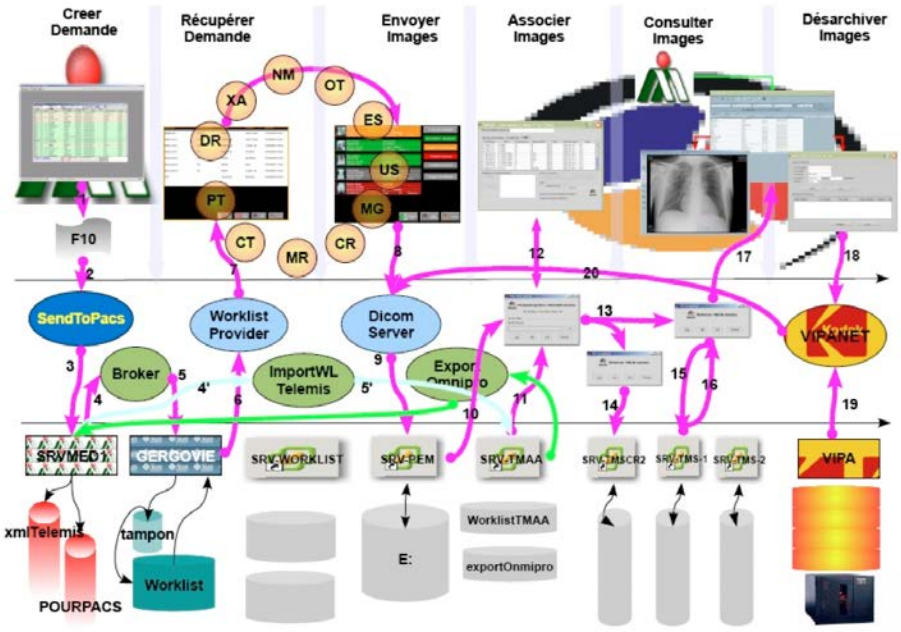

Figure 1.1 – Représentation du *workflow* de patients à Mont-Godinne [4]

Le projet IFLEC a aussi lancé l'analyse détaillée des composants afin de comprendre les interactions et l'échange d'informations entre ceux-ci. L'utilisation de standards de communication et de stockage pourrait permettre un meilleur interfaçage entre les différents logiciels mais surtout garantir un meilleur contrôle sur l'évolution et la maintenance de l'architecture mise en place actuellement. De tels standards sont cruciaux dans une architecture regroupant des dizaines de logiciels. Un avantage indéniable est également la possibilité de généraliser les échanges de données avec n'importe quel centre hospitalier [4].

1.1.2 Dossier Patient Informatisé

Le dossier patient informatisé ou DPI [1] permet de centraliser les données des patients de manière électronique. Les données qu'il contient sont collectées par différents acteurs, le plus souvent le médecin généraliste mais pas seulement. En Belgique, l'origine du DPI remonte à 1997 où un groupe d'expert du Ministère des soins de santé a commencé à constituer les éléments d'analyse. À cette fin, la structure logique du DPI est importante [5].

Les progrès technologiques de ces dernières années ont considérablement fait croître l'importance du DPI dans le système de santé belge. Les formes sont diverses et varient du dossier partagé au sein d'un établissement au dossier partagé par plusieurs institutions. Des programmes informatiques permettent de consolider chaque

1. En anglais, le dossier patient informatisé se nomme *Electronic Medical Record* ou EMR

jour le DPI par l'ajout d'informations mais aussi de communiquer les données avec d'autres hôpitaux. De ce fait, le dossier patient est devenu un élément primordial dans les décisions médicales : elles sont prises suivant les antécédents des patients se trouvant dans le DPI [4].

Outre le fait d'enregistrer et communiquer l'information contenue dans le dossier patient, retrouver une donnée ou une information n'est pas toujours facile. Les logiciels de gestion ne proposent pas de fonctions de recherche d'informations dans le DPI. Cette nécessité est parfois vitale car la taille des données concernant chaque patient augmente toujours plus [4]. Dans un milieu où chaque seconde perdue a de lourdes conséquences, la création de logiciels *nouvelle génération* est essentielle.

Afin de retrouver facilement les informations désirées, une restructuration de celles-ci est inévitable. Dans les logiciels médicaux, les informations sont faiblement structurées excepté pour les données démographiques, les prises en charge hospitalières, le codage des diagnostics et des actes et enfin les résultats d'analyse biologiques [4]. Pour le reste, il faut se contenter de données textuelles c'est-à-dire non structurées. Pour élaborer de nouvelles applications, il fallait revoir le système d'enregistrement des données. Cela passe par l'utilisation d'un standard.

1.2 Les Standards Existants

Plusieurs standards existent à l'heure actuelle pour stocker les données des patients mais également pour échanger ces données entre différentes institutions. Les sous-sections suivantes présentent les différents standards qui concernent les études cliniques plus généralement.

1.2.1 HL7

Le *Health Level Seven*, ou plus communément appelé le HL7 [6], est une organisation à but non lucratif fondée en 1987. Cette organisation regroupe des professionnels de la santé à travers le monde, même si elle a été initialement créée pour les États-Unis. Elle s'occupe de fournir des standards mondiaux pour le milieu de la santé comme des standards d'échanges électroniques de données médicales. Plusieurs nations, dont la Belgique, entrent en jeu pour l'élaboration des standards. D'ailleurs, l'organisation est reconnue et accréditée par l'ANSI (*American National Standards Institute*) et l'ISO (*International Organization for Standardization*) [6].

Le nom HL7 fait autant référence aux standards qu'à l'organisation elle-même. Les standards HL7 sont disponibles en plusieurs versions. Les versions les plus utilisées sont néanmoins les versions 2.x alors qu'une minorité utilise la version 3.0 pourtant développée depuis 1995 et publiée en 2005 [7]. La figure 1.2 montre comment est répartie l'utilisation des différentes versions de HL7 [8].

Dans les versions 2.x, il y avait de multiples variations possibles. Les parties désirant communiquer devaient au préalable s'accorder entre elles sur les messages avant de pouvoir commencer les transmissions. Alors que HL7 désire standardiser au maximum et transmettre de l'information facilement, les versions 2.x ne sont pas

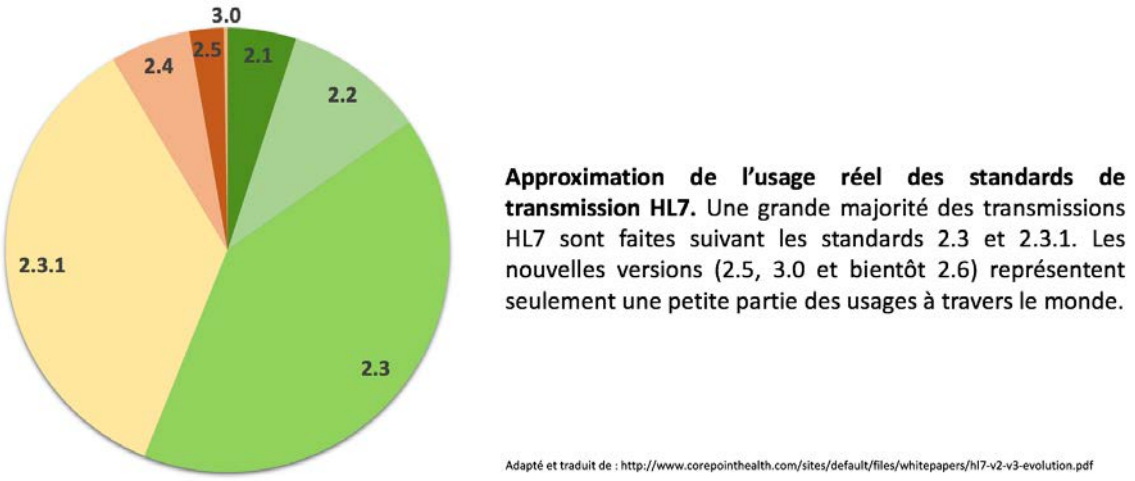

Figure 1.2 – Répartition des versions de HL7 (2006)

avantageuses en terme d'interopérabilité. Les négociations au préalable entre les interlocuteurs ralentit et décourage les échanges entre institutions car il est nécessaire de réaliser un travail de transformation des données lors de la réception de messages.

Pour combler le manque d'interopérabilité, la version 3.0 a vu le jour en 2005 après 10 ans de développement. Cette version est beaucoup plus formalisée que la précédente et les messages sont maintenant typés. Cela est dû au respect d'un autre standard appelé RIM (*Reference Information Model*) [9]. RIM est un modèle qui représente le domaine (i.e. les données) clinique et les cycles de vie d'un message. Tout message HL7, depuis sa version 3.0, se base sur ce modèle.

Alors que les versions 2.x n'autorisent que la transmission de messages peu structurés, c'est-à-dire où les données transmises sont du texte divisé par des séparateurs, la version 3.0 est beaucoup plus stricte. Cette nouvelle version repose sur la technologie XML (*eXtensible Markup Language*). Cela rend la rétrocompatibilité quasi nulle entre les versions 2 et 3. L'utilisation de versions différentes pour les professionnels du secteur de la santé reste un frein dans l'utilisation massive de HL7. Bien que la version 3 soit beaucoup plus avantageuse en terme de formalisme, ceux qui utilisent cette dernière version sont souvent obligés d'utiliser une version 2 en supplément pour pouvoir communiquer, étant donné l'incompatibilité des deux versions.

La standardisation de HL7 se fait via l'utilisation de fichiers XML. Les messages doivent alors respecter une syntaxe bien définie pour éviter la non compréhension par une machine. La capacité, pour une machine, à pouvoir comprendre et traiter les données dans leur intégralité est aussi une force de la version 3.0. La structure est définie par le CDA (*Clinical Document Architecture*) [10]. Il fournit une structure à adopter pour les messages HL7 transmis par fichiers XML. La figure 1.3 illustre la structuration des éléments contenus dans le CDA.

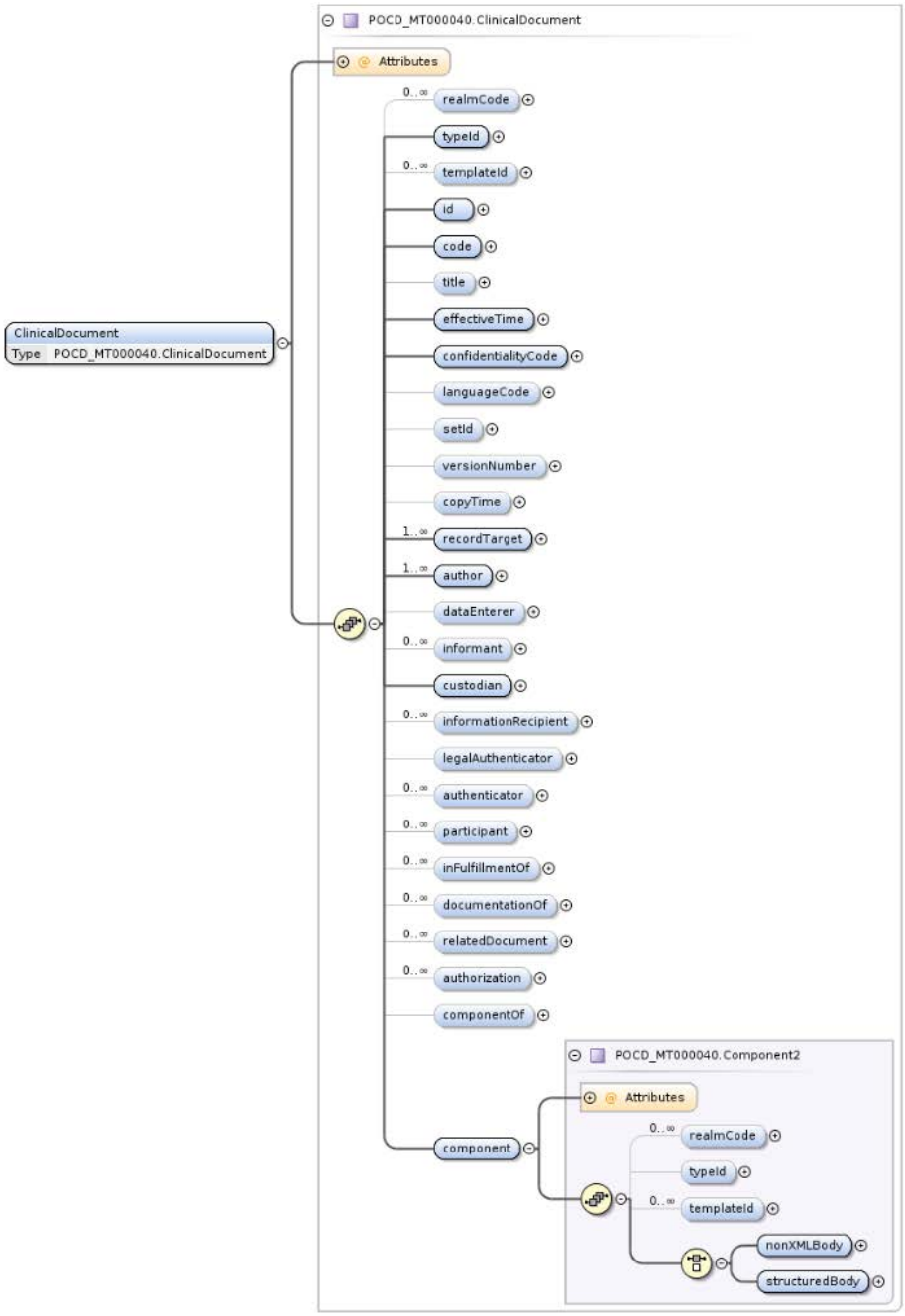

Figure 1.3 – Les différents éléments dans le CDA [7]

1.2.2 KMEHR

KMEHR ou *Kind Messages for Electronic Healthcare Record* est un standard sur les données médicales proposé par la Belgique en 2002. Ce standard est actuellement maintenu par la plateforme gouvernementale eHealth et utilisé sur le Réseau Santé Wallon. Il s'agit d'une implémentation de la quatrième recommandation de la Belgian Health Telematics Commission pour créer un échange de données médicales structurées [11]. Il est aujourd'hui et depuis quelques années le standard technique pour échanger les informations entre les centres hospitaliers, les médecins et autres praticiens [12].

« *Le site web KMEHR documente le standard belge. Dans une deuxième et une troisième phase, le projet procurera au secteur médical un outil permettant de valider des applications logicielles médicales ainsi qu'une méthode de travail et un rapport standardisés pour les analyses en laboratoire* » [12]. Pour le moment, KMEHR veille à ce que les informations parviennent rapidement à la bonne personne par le biais de messages XML sécurisés. Ces messages expriment des transactions médicales. Des tables de références qui contiennent des valeurs pour décrire des situations font aussi partie de KMEHR.

Le standard KMEHR s'articule autour de trois éléments [7] :

— Un schéma XML (ou XSD) qui définit la structure et la grammaire des documents XML à envoyer. Le format du message KMEHR est structuré suivant ce schéma et ce dernier précise l'emplacement des éléments contenus dans le message.
— Des transactions médicales qui indiquent le type de message à envoyer comme le dossier patient, le résumé des médicaments ou encore les images médicales.
— Des tables de références qui contiennent des valeurs numériques qui font référence à des situations bien précises. Par exemple, il existe un code particulier pour identifier une grossesse.

Les documents XML de KMEHR sont donc structurés selon une grammaire bien particulière. Chaque document contient un *Header* et au moins une *Transaction* contenue dans un *Folder*. Les figures 1.4 et 1.5 illustrent la structuration d'un message KMEHR [13].

```
<?xml version="1.0" encoding="UTF-8"?>
<kmehrmessage ... link to XSchema ... >
   <header>
   </header>
   <folder>
      <patient> ... identification of the patient ... </patient>
      <transaction>
         <item> ... type of encounter ... </item>
         <item> ... date of encounter ... </item>
         <heading>
            <item> ... diagnosis ... </item>
         </heading>
         <heading>
            <item> ... drug delivery ... </item>
            <item> ... drug prescription ...</item>
         </heading>
      </transaction>
   </folder>
</kmehrmessage>
```

Figure 1.4 – Représentation d'un message KMEHR en XML

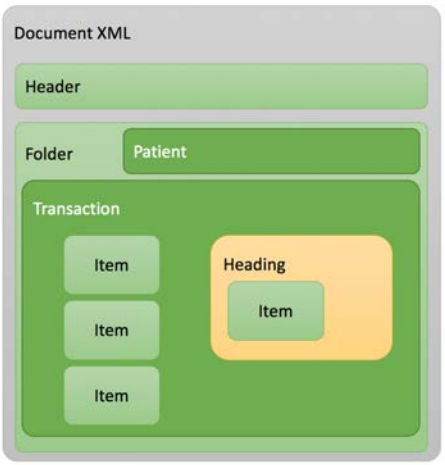

Figure 1.5 – Schéma d'un message KMEHR

Le standard KMEHR permet d'utiliser plusieurs « niveaux » de standardisations dans un document. « *Plusieurs possibilités existent pour formater un document KMEHR : le document le plus simple est très peu structuré et sert juste au transport d'un texte brut incompréhensible par un ordinateur, tandis que le plus perfectionné*

exprime des données de manière entièrement interprétable par un système automatique. Les quatre niveaux possibles sont les suivants » [7] :

— Niveau 1 : Un code de transaction est incoporé dans le message KMEHR pour déterminer son type (dossier patient, instruction de soins, etc.). Par contre, le reste du message est du texte libre donc non interprétable.
— Niveau 2 : Un peu plus structuré que le niveau 1, ce type de message contient une partie codifiée et interprétable dans la partie *Header*. Les informations sur le patient et sur l'émetteur du message peuvent être comprises par une machine. Le reste du document est toujours du simple texte.
— Niveau 3 : Pour structurer un peu plus le message, les éléments contenus dans *Folder* sont codés mais leur contenu, lui, ne l'est pas.
— Niveau 4 : Le niveau 4 indique que tous les éléments sont structurés même les informations contenues dans *Folder*. Une machine peut tout à fait interpréter l'entièreté du message KMEHR.

Une précision est à apporter sur un genre de transaction de KMEHR : le sumEHR (*Summarised Electronic Health Record*). Ce type de transaction permet de partager un résumé sur le patient. L'information est utilisée par le personnel hospitalier pour comprendre les besoins du patient ou son état de santé général. De ce fait, sumEHR est approximativement similaire au dossier patient. Il faut également préciser que KMEHR fournit les outils nécessaires pour diffuser les informations aux autres institutions utilisant ce standard. En effet, des échanges sont possibles grâce à l'utilisation de *Web Services* mis à disposition par la plateforme eHealth [11].

1.2.3 vMR

Un autre modèle sur lequel il faut s'attarder est celui de vMR ou *Virtual Medical Record* [14]. Il s'agit d'un modèle créé par l'organisation HL7 visant à représenter les informations médicales d'une manière plus standardisée. vMR vise à rendre les échanges d'informations entre différents logiciels, que peuvent être les systèmes d'information « maisons » ou les systèmes d'aide à la décision au niveau médical (CDSS - *Clinical Decision Support System*), beaucoup plus aisés.

Sans vraiment de standard pour représenter les données au sein d'une institution, les efforts de transformation de celles-ci pour leur transmission via des procédures standardisées ou non sont conséquents. Il faut donc trouver un moyen de standardiser l'information. Les limitations dans l'implémentation de systèmes complexes au sein des hôpitaux sont surtout dus à une utilisation de standards différents selon l'établissement et au manque de représentation standardisée des données médicales et des terminologies (i.e. dictionnaires). C'est dans cette optique que le standard vMR a été créé. Le but ultime de l'organisation HL7 est de donner, via le modèle vMR, une représentation standardisée de l'information en entrée et en sortie. Ceci signifie un enregistrement tel quel des données mais aussi une transmission telle quelle sans modification préalable. [15].

Le modèle contient plus d'une centaine d'éléments qui définissent des entités du domaine médical. vMR autorise la représentation de grandes quantités d'informa-

tions concernant le patient. Durant l'implémentation de ce modèle, 22 établissements provenant de 4 pays différents ont participé. Le modèle vMR est en passe de devenir le standard de représentation de la connaissance du milieu médical pour contrer le manque d'interopérabilité des autres standards [15].

Le modèle fournit une représentation complète du milieu hospitalier à un certain niveau d'abstraction. Le projet de définition des entités a débuté en 2010. La dernière version du modèle date de janvier 2014. L'analyse se base sur cette dernière version [16].

La documentation de vMR fournit sur soixante pages la description de chaque entité. Pour plus de lisibilité, le modèle a été découpé par HL7 suivant 19 grands concepts. La figure 1.6 illustre le concept de relevé clinique (*Clinical Statement*) tandis que la figure 1.7 représente le concept général de vMR.

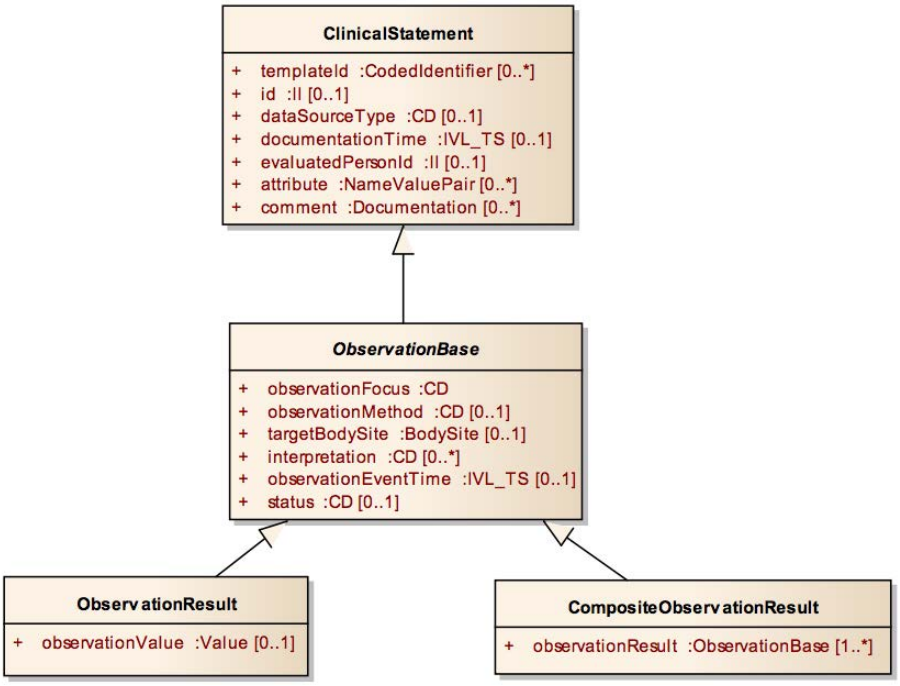

Figure 1.6 – Représentation du concept de *Clinical Statement* [16]

La figure 1.7 est le point de départ du modèle. Chaque entité qui y est représentée est définie dans un autre schéma. C'est le cas notamment de *Clinical Statement* qui se voit précisé à la figure 1.6.

La construction des entités est réalisée en UML (*Unified Modeling Language*) [16] et représente le monde médical sous forme de concepts liés entre eux par des relations hiérarchiques. Ces relations sont particulières et sont nommées *relations is-a*. De telles relations englobent les notions de *sur-type* et de *sous-type*. Par exemple, dans le cas de *Clinical Statement*, *ClinicalStatement* est sur-type de *ObservationBase*. La lecture peut également se faire dans le sens contraire : *ObservationBase* est sous-type de *ClinicalStatement*. Cette construction est présente dans l'ensemble du modèle vMR.

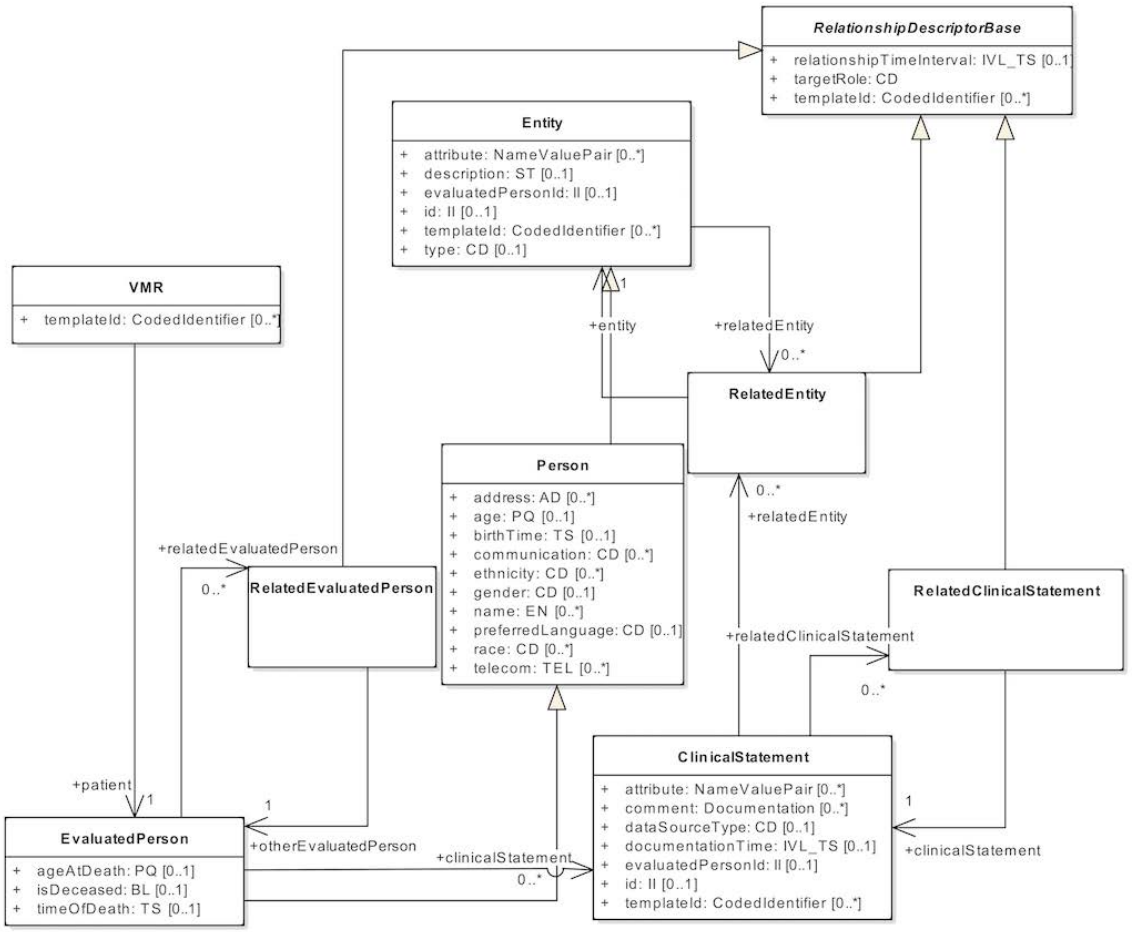

Figure 1.7 – Représentation du concept général de vMR [16]

Alors que vMR voudrait s'imposer en tant que standard, aucune implémentation du modèle n'est fournie. A l'heure actuelle, vMR reste un modèle, un schéma de représentation du domaine médical. Pour devenir le standard par excellence, l'organisation HL7 devrait fournir une base de données qui repose sur vMR. Ce n'est pas encore le cas actuellement. Il faut noter que le modèle vMR est toujours en phase de test et que des mises à jour et/ou extensions sont possibles. Une nouvelle parution est prévue pour début 2016 [14].

1.2.4 openEHR

openEHR [17] est une communauté virtuelle qui travaille sur l'interopérabilité des logiciels dans le domaine de la santé. Elle se concentre principalement sur les Dossiers Patient Informatisés (DPI ou EHR en anglais). La fondation openEHR a publié un ensemble de spécifications concernant la gestion, le stockage et l'échange de messages électroniques.

La caractéristique de openEHR est qu'une définition du domaine médical à deux niveaux est fournie : d'une part, le modèle de référence et d'autre part, les archétypes.

« *Le modèle de référence représente les caractéristiques générales des éléments de données d'un dossier médical. Ce modèle définit un ensemble de classes qui constituent les principaux blocs et sections que l'on retrouve dans un dossier électronique (ex : les classes Observation, Evalution, Instruction et Action). Le modèle de référence est donc très générique et insuffisant pour définir, de manière plus détaillée, les éléments de données nécessaires au domaine médical (par exemple, les éléments de données concernant la tension artérielle)* » [18].

Pour étendre et détailler davantage les éléments propres au domaine médical, il a fallu définir les archétypes. Ceux-ci sont des structures d'information qui contiennent des données sur les patients. Ainsi, les structures sont composées de classes du modèle de référence. Des contraintes sont ajoutées par l'intermédiaire des archétypes. Le langage ADL (*Archetype Definition Language*) exprime ces contraintes. Un exemple de contrainte est celui-ci : *la valeur minimale ou maximale autorisée d'un élément de données* [18]. En résumé, les archétypes sont utilisés en tant que représentation d'un concept spécifique comme la pression sanguine. openEHR définit des états, des événements, des protocoles et des données sur ces concepts à travers les archétypes.

Les archétypes ne sont malheureusement pas utilisables tels quels car ils sont beaucoup trop précis pour un contexte donné. Généralement, chaque établissement de soins travaille sur la modification et l'utilisation d'une partie des archétypes. Certains experts estiment qu'il faudrait disposer de plusieurs centaines de milliers d'archétypes pour couvrir la totalité du domaine médical. Implémenter chaque archétype est une utopie pour un hôpital [18]. La figure 1.8 illustre l'archétype sur la pression sanguine.

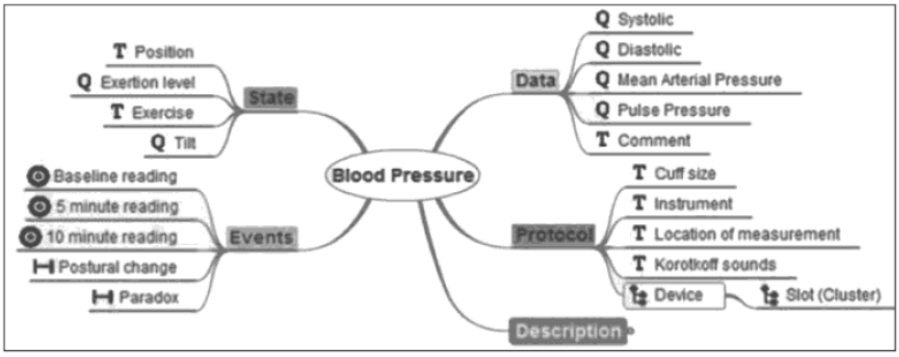

Figure 1.8 – Représentation de l'archétype sur la pression sanguine [17]

1.2.5 Autres Standards

Moins importants que les standards présentés ci-dessus, d'autres standards existent néanmoins. Parmi ceux-ci se trouvent HISA, BRIDG, EDIFact et CDISC.

HISA La *Health Information System Architecture* ou HISA est une norme qui vise à définir une série de structures pour permettre la réalisation de logiciels médicaux. Cette norme a été conçue par une organisation européenne à partir de 1989. En 2009, HISA a été adoptée par l'Organisme de Normalisation ISO (*ISO 12967*). La norme fournit des orientations sur l'architecture des logiciels médicaux ainsi que sur l'intégration de plusieurs logiciels entre eux [19].

BRIDG Élaboré par le *Biomedical Research Integrated Domain Group* (BRIDG), le modèle BRIDG vise à produire des documents pour la modélisation des données collectées par la recherche clinique. Les échanges, les processus, les caractéristiques biologiques et les procédures sont modélisés à travers des diagrammes [20].

EDIFact Basé sur l'EDI (*Échange de Données Informatisées*), la variante EDIFact est une norme internationale utilisée pour échanger des données dans l'Administration, le Commerce et le Transport. La norme définit un ensemble de règles très structurées pour permettre une communication entre des systèmes d'information indépendants [21].

CDISC Les standards CDISC, élaborés par l'organisation du même nom, visent à uniformiser la récupération, les échanges et la soumission des données médicales. A l'heure actuelle, les standards sont toujours en développement pour supporter l'ensemble des processus des études cliniques, allant de la production de protocoles d'études à la création de rapports [22]. Ces standards ont déjà fait l'objet d'une analyse profonde en 2014 par un étudiant de l'Université de Namur [3]. Malgré la présence de ces standards au niveau des études cliniques, l'analyse n'est pas concluante. La solution proposée par CDISC est trop spécifique et personnalisée, ce qui n'est pas un avantage pour une solution standardisée [3]. De ce fait, une nouvelle analyse n'a pas été effectuée et le standard a été écarté.

1.3 Avantages et Inconvénients des Modèles

Cette dernière section vise à résumer la précédente en exprimant les différents avantages et inconvénients des standards présentés antérieurement. Un tableau récapitulatif est présenté au tableau 1.9. Il présente, pour chaque standard, son type, son utilisation, sa couverture du domaine (au niveau des études cliniques), sa base de conception, son stade d'implémentation et son type de licence.

	Standard	Utilisation	Couverture du domaine	Base de conception	Niveau du standard	Licence
HL7	Communication	USA et Europe	Large	XML	Implémenté	Payant
KMEHR	Communication	Belgique	Faible	XML	Implémenté	Gratuit
vMR	Stockage	Globale	Large	UML	Théorie	Gratuit
openEHR	Stockage et Communication	Globale	Faible	UML	Implémenté	Gratuit
HISA	Création de logiciels	Europe	-	-	Implémenté	Gratuit
BRIDG	Communication	USA	Large	UML	Implémenté	Gratuit
EDIFact	Communication	USA	Moyenne	EDI	Implémenté	Gratuit

Figure 1.9 – Tableau récapitulatif des standards présentés dans ce chapitre

1.4 Logiciels Associés

Les standards de stockage et de communication, présentés au travers de ce chapitre, ont pu servir de base à certains outils. Développés par des entreprises ou libres de droits, ces logiciels sont cités ci-dessous. Par ailleurs, le lecteur est invité à se rendre à l'annexe A pour découvrir les interfaces graphiques des différents programmes.

- Soarian : créé par Siemens, ce logiciel comporte entre autres des fonctionnalités de création de flux (comme pour le déroulement d'une étude clinique par exemple), de gestion de patients et de commande de médicaments. Il est surtout connu pour son interopérabilité avec d'autres logiciels au niveau de la communication et de l'intégration [23].

- CareConnect : Cet outil permet l'interaction avec tous les prestataires de soins et gère les données patients d'une façon beaucoup plus conviviale que d'autres logiciels. Basés sur le *cloud*, toutes les données sont stockées sur un serveur externe. De ce fait, le logiciel autorise également une utilisation hors ligne [24].

- OpenCDS : Ce logiciel (libre) fournit un standard dans les interfaces pour le domaine de la santé. Ses fonctionnalités visent tout particulièrement à générer des conclusions sur des données patients, gérer les terminologies et créer des requêtes personnalisées sur les données médicales [25].

- InteliChart : Il s'agit d'un portail qu'une institution peut mettre à disposition des patients. Il permet accessoirement la gestion des rendez-vous, le remplissage de formulaires en ligne, le rappel de prises de médicaments et l'affichage de résultats de laboratoire [26].

Chapitre 2

Ontologies

Afin de continuer le cheminement vers un nouveau genre de logiciels cliniques, ce chapitre traite des ontologies. Pour commencer, la première section présentera le concept général en étudiant de près les composants clés et les langages concernés. Ensuite, une section sera consacrée aux logiciels associés aux ontologies. Enfin, une dernière section traitera des ontologies et leur importance dans le milieu médical.

2.1 Présentation

En guise d'introduction au chapitre, cette première section vise à donner une vision plus claire des ontologies au lecteur.

2.1.1 Définition de l'Ontologie

La définition de l'ontologie varie selon le domaine. A l'origine, le mot *ontologie* est un concept philosophique pour désigner l'étude de l'être et de ses propriétés générales en tant qu'être [27, 28]. Le même concept au niveau informatique serait de dire qu'il s'agit d'étudier un domaine et de définir ses propriétés. Une ontologie, au sens informatique du terme, est en fait une structuration des concepts d'un domaine bien particulier qui vise à offrir une base solide pour se représenter le monde (i.e. le domaine en question) [29]. Une ontologie capture la totalité de la connaissance d'un concept en identifiant les éléments utiles ainsi que les relations entre ces éléments.

2.1.2 Composants d'une Ontologie

Une ontologie possède différents composants qui permettent, ensemble, de définir un domaine particulier. Ces composants sont les « classes », les « individus », les « propriétés » et les « règles ». En combinant ces quatre composants, une ontologie peut être créée [30, 31].

Classes Une classe (ou un *concept* dans certains ouvrages) désigne un objet ou un groupe d'objets, une idée ou encore une notion abstraite [30]. Elle regroupe les objets du domaine en cours d'élaboration et qui partagent certaines caractéristiques. Il s'agit de l'élément le plus utilisé dans la création d'une ontologie. Il est possible de définir également la notion de sous-ensemble. Dès lors, il s'agit de décrire le lien entre une classe et sa sous-classe. Une hiérarchie peut être définie de cette manière entre les concepts [31]. La classe de base est appelée *Thing* et contient toutes les classes définies. Un exemple simple [1] illustre la notion de classes dans la figure 2.1.

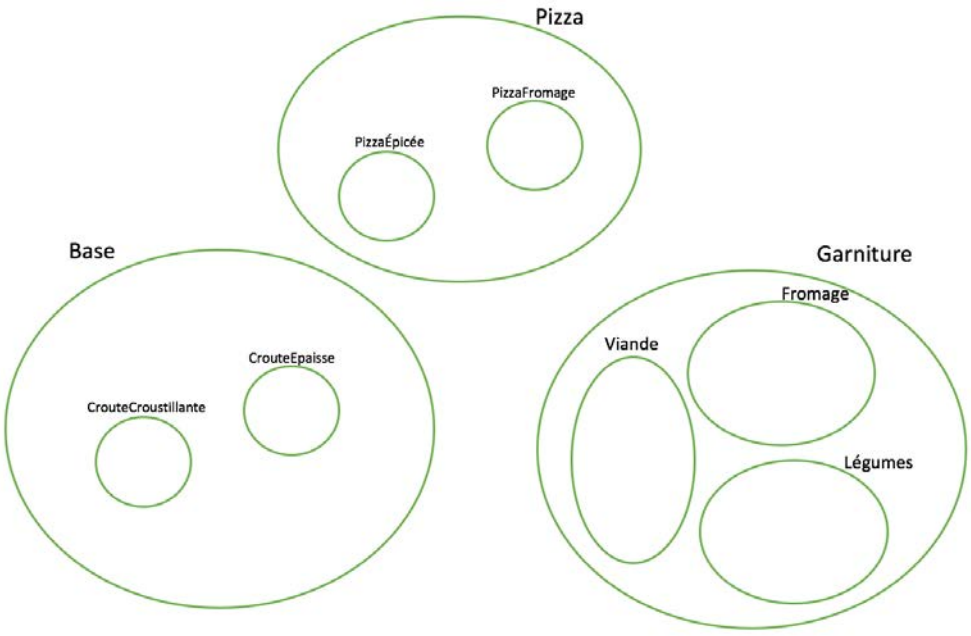

Figure 2.1 – Exemple de classes dans une ontologie

Individus Les individus, appelés aussi *instances*, représentent les éléments clés pour l'ontologie. Il s'agit d'objets concrets peuplant les classes [31]. Ainsi, pour continuer l'exemple précédent, la figure 2.2 ajoute la notion d'individus à la représentation de l'ontologie.

1. Cet exemple sur les pizzas s'inspire du document explicatif sur Protégé [31]

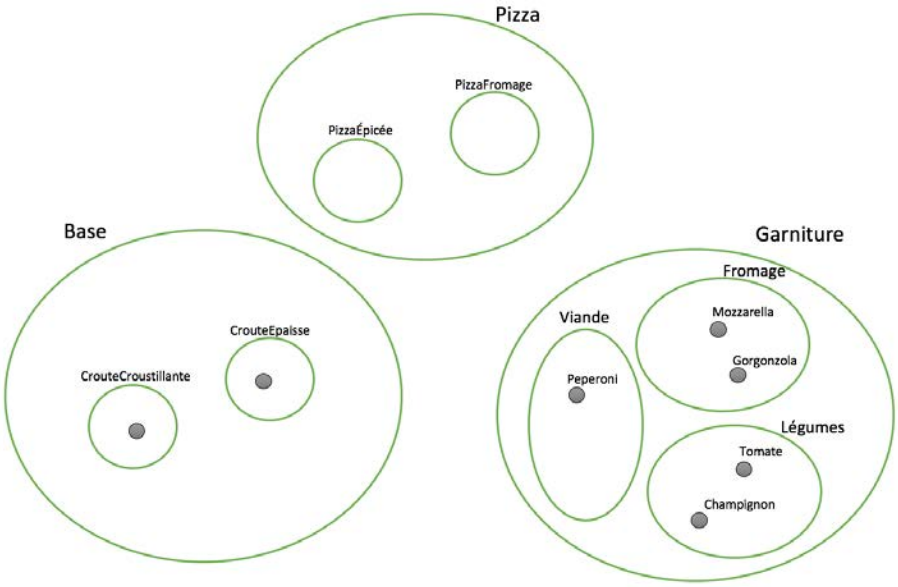

Figure 2.2 – Exemple d'individus dans une ontologie

Propriétés Les propriétés expriment les relations entre deux individus ou entre deux classes [31]. Les propriétés inverses sont possibles. Par exemple, la propriété initiale « a comme base » possède comme propriété inverse « est la base de ». En outre, les notions de transitivité et de symétrie peuvent être appliquées aux propriétés. Un exemple de quelques propriétés est disponible à la figure 2.3.

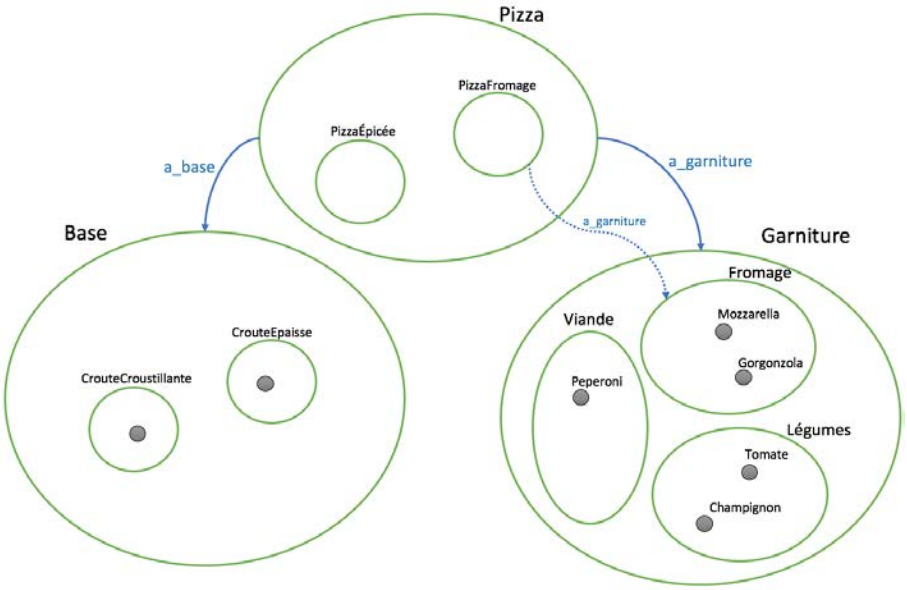

Figure 2.3 – Exemple de propriétés dans une ontologie

Indépendamment des propriétés qui lient deux individus ou deux classes, des propriétés supplémentaires sur des classes peuvent être créées en combinant les propriétés existantes avec des individus. Ainsi, des classes d'équivalence peuvent être établies. Les classes d'équivalence sont par exemple utilisées pour dire qu'une pizza au fromage est une pizza dont la garniture est le fromage. Il faut alors écrire ceci [2] :

```
PizzaFromage EquivalentTo Pizza and (a_garniture some Fromage)
```

Règles Un quatrième composant intervient dans la création des ontologies. Il s'agit des règles (ou également appelés axiomes) [30]. Ces règles sont des expressions qui sont toujours vraies quels que soient les éléments pris en compte. Elles décrivent, dans un langage logique, la sémantique des relations et des classes. Par exemple, sur la figure 2.3, les deux classes *Viande* et *Fromage* sont disjointes.

Afin de conclure cette sous-section, une illustration de l'ontologie utilisée dans les exemples précédents est décrite à la figure 2.4.

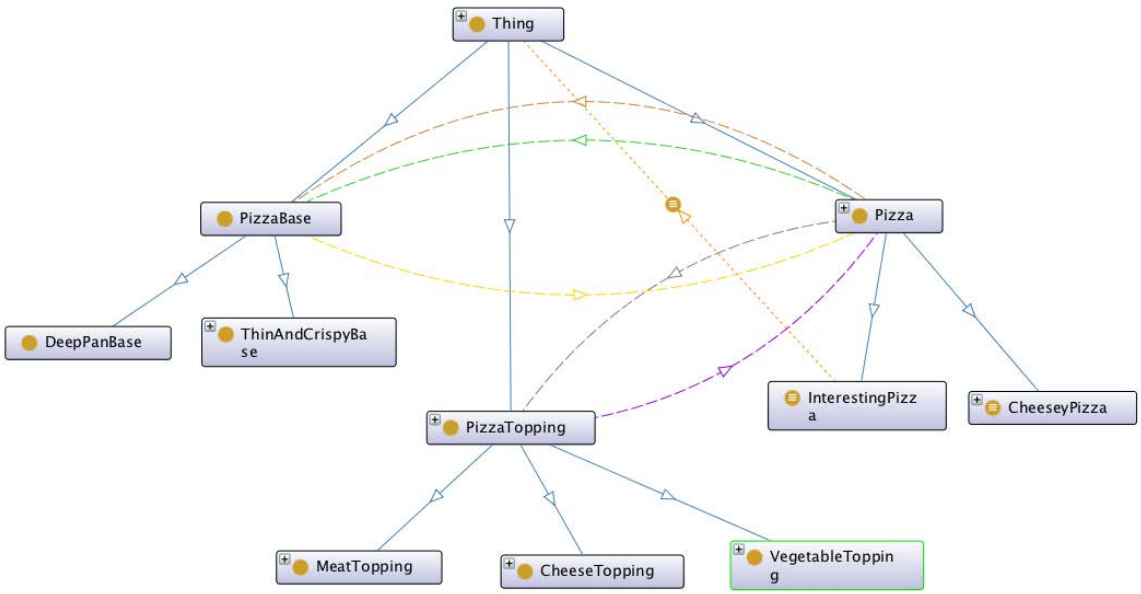

Figure 2.4 – Description d'une ontologie sur les pizzas

2. Il s'agit d'une anticipation par rapport aux langages utilisés pour créer des ontologies. La sémantique d'écriture est propre au logiciel Protégé. Pour plus de détails, voir la sous-section 2.1.3 sur les langages et la section 2.2 sur les logiciels concernés.

2.1.3 Langages

C'est dans les années 80 que les langages pour représenter les ontologies et les connaissances ont vu le jour. Depuis lors, de nombreux langages ont fait leur apparition. Ceux-ci peuvent se baser sur trois formalismes de représentation : les logiques de descriptions, les frames et les graphes conceptuels. Pour créer des ontologies les logiques de description sont les plus adaptées car leur capacité à raisonner automatiquement sont leur plus grand avantage [30].

Récemment, quelques langages ont émergé pour la construction d'ontologies. Ceux-ci sont RDF, RDFS, OWL, OIL, XOL, SHOE ou encore DAML+OIL. Dans les paragraphes suivants, seuls quelques langages intéressants seront présentés. La figure 2.5 fait la synthèse des langages en les classant sous forme pyramidale.

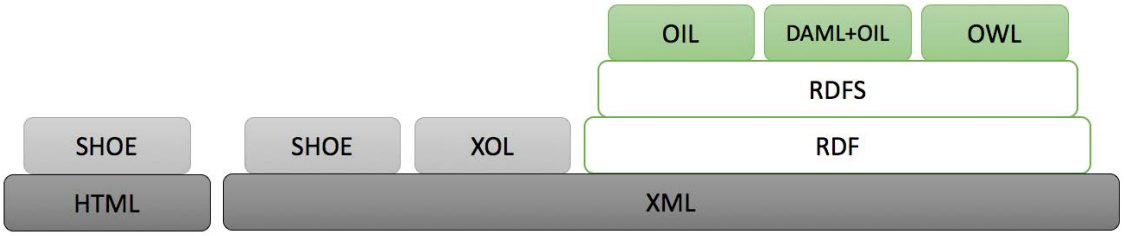

Figure 2.5 – Classification pyramidale des langages d'ontologies [30]

Sur la figure précédente, les langages inférieurs constituent la base des langages supérieurs. SHOE est basé sur XML ou sur le standard HTML suivant la version utilisée. XOL et RDF (ainsi que son extension RDFS) sont eux basés uniquement sur la syntaxe XML. Plus haut dans la pyramide se trouve OWL. Il s'agit du langage le plus utilisé et dispose de trois extensions : OWL Lite, OWL DL et OWL Full. Les deux derniers langages, OIL et DAML+OIL, sont eux aussi basés sur RDF(S).

RDF et RDFS Afin d'échanger des données sur le web de manière standardisée, le W3C a développé le modèle RDF, ou *Resource Description Framework* [32]. Le modèle permet de représenter des entités et des propriétés d'un domaine bien spécifique et de créer ainsi une connaissance. Un document RDF est un document proche du document XML mais est basé sur des triplets « Sujet , Prédicat, Objet » [30]. Alors que le document s'écrit de manière textuelle, la représentation schématique se fait via un graphe [33]. Ce genre de graphe est beaucoup plus expressif et lisible pour l'être humain.

Dans un graphe RDF, le *sujet* (ou la ressource) représente une entité qui est décrite. Une identification de chaque entité permet de garantir l'unicité. Cela est possible par l'utilisation d'URI (*Uniform Resource Identifier*) [33]. Une URI est par exemple : `http://www.exemple.org/pizza/garniture`. Le *prédicat* sert à décrire la relation entre les entités. Il est également identifié par une URI. Enfin, l'*objet* représente une autre entité ou ressource [30].

L'extension de RDF est le RDFS (*RDF Schema*) et offre la possibilité de spécifier un schéma directeur. Ce schéma propose une spécification des concepts, des taxonomies et des relations binaires [30]. C'est grâce au schéma que les contraintes sur le domaine sont précisées. Par exemple, les valeurs que peuvent prendre les propriétés ou les hiérarchies entre les entités sont décrites à partir de RDFS [34].

OWL Le langage OWL (*Web Ontolgy Language*) [35] est basé sur le modèle RDF(S) et a également été conçu par le W3C. Ce langage est, au contraire de RDF, beaucoup plus expressif dans son vocabulaire et permet donc une description beaucoup plus pointilleuse des entités du domaine que l'on cherche à représenter. Il est recommandé par le W3C d'utiliser ce langage pour développer des ontologies [35]. Trois sous-langages font partie de OWL et la figure 2.6 illustre leur organisation.

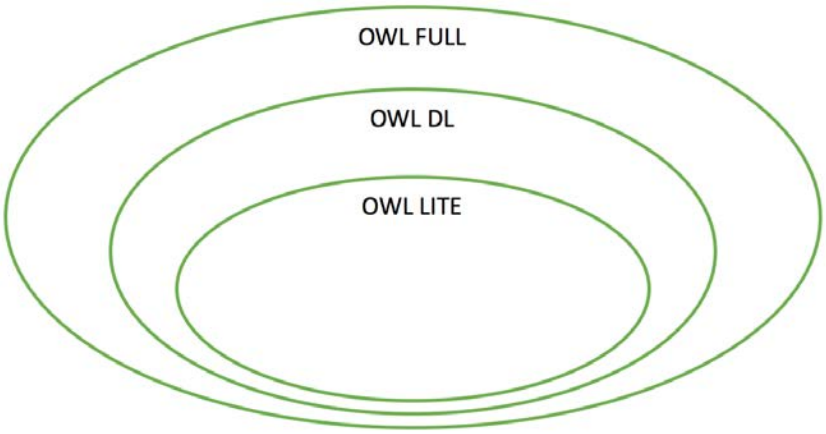

Figure 2.6 – Classification des langages OWL

1. **OWL Lite :** Le langage le plus simple est OWL Lite. Il est aussi celui qui offre le moins d'expressivité [36]. Malgré tout, le langage répond aux besoins primaires d'une modélisation en donnant la possibilité de faire des hiérarchies dans les classifications, de créer des propriétés inverses et de définir des contraintes simples limitées à la cardinalité [0-1]. Le fait d'avoir une complexité formelle faible lui procure un avantage par rapport aux deux autres sous-langages et les requêtes s'effectuent plus rapidement [30, 34].

2. **OWL DL :** Langage intermédiaire entre OWL Lite et Full, OWL DL est également le langage le plus répendu et le plus utilisé dans la création d'ontologies. Il offre une expressivité maximale et garantit la complétude et la décidabilité [30, 36]. Cela signifie que toutes les conclusions sont calculables [3] en un temps raisonnablement court. OWL DL (DL pour *Description Logic*) se base donc sur la logique descriptive. Ce genre de logique est un sous-ensemble de la logique du premier ordre et propose de définir de manière formelle des concepts d'un domaine particulier [34].

3. Dire que des conclusions sont calculables revient à dire qu'un ordinateur peut effectuer des traitements sur les éléments et retourner une valeur à l'utilisateur dans un temps déterminé et fini.

3. **OWL Full :** Le dernier langage, OWL Full, est celui qui offre l'expressivité la plus complète car il n'a aucune restriction sur le vocabulaire utilisé. Comme il est le plus complet, il exploite le langage RDF au maximum et est donc compatible avec celui-ci [34]. Cependant, OWL Full ne possède pas les propriétés de décidabilité, au contraire de OWL DL. Par ailleurs, aucune implémentation intégralement réalisée n'existe à ce jour [30].

OIL et DAML+OIL OIL (*Ontology Inference Layer*) a été conçu par le W3C début des années 2000 et utilise les fonctionnalités de base de RDF. Sa conception avait pour but la création d'ontologies uniquement [37]. Le langage a évoluer pour devenir DAML+OIL, une combinaison de deux langages. Cette évolution majeure a permis d'ajouter une approche orientée *objet* au modèle. Malgré cela, le projet a pris fin en 2006 et le langage a clairement été supplanté par OWL [7, 37].

Autres langages Pour expliquer l'ontologie sur les pizzas, il a été dit précédemment qu'il y avait une notation particulière pour créer des propriétés sur des entités. Ce type de notation est propre à certains logiciels comme par exemple Protégé. Le logiciel se charge ensuite de réaliser la transformation vers le langage souhaité (OWL, RDF, etc.)

2.1.4 Web Sémantique

Le web sémantique, appelé aussi le web 3.0 ou encore le web des données, devient une préoccupation grandissante au fil des années. Il vise à devenir la prochaine évolution majeure en terme de connexion avec l'information [38]. Une nouvelle vision du web tend à être apportée avec le web sémantique : passer d'une vision orientée *Documents* vers une vision *Connaissance* en éliminant l'ambiguïté [33]. A l'heure actuelle, l'information se trouvant sur la toile n'est pas organisée et est souvent représentée de manière textuelle. Imaginez un web où l'information est structurée et où des informations connexes sont disponibles à la demande et surtout traitables par un ordinateur. C'est ce que vise à réaliser le web sémantique par l'utilisation massive d'ontologies.

Avant l'apparition de l'Internet et du web, les liens entre les documents écrits (livres, dictionnaires, encyclopédies, etc.) étaient rendus possibles grâce aux citations. L'inconvénient principal était l'incapacité à parcourir le document cité si ce n'était d'avoir l'exemplaire en question proche de soi. Ensuite, le web 1.0 est apparu avec ses liens hypertextes. Un clic sur un lien ouvrait directement la page correspondante. L'information peut être parcourue de proche en proche. L'évolution vers le web 2.0 amène les réseaux sociaux où l'information est transmise directement par l'utilisateur. L'interaction est le point central. Finalement, le web sémantique ne vise plus à faire des liens entre les documents mais à faire des liens entre les faits. Il n'a plus lieu de se contenter d'un document particulier. L'information est construite suivant les faits se trouvant sur des documents différents [38].

La figure 2.7 illustre l'évolution du web et sa gestion de l'information.

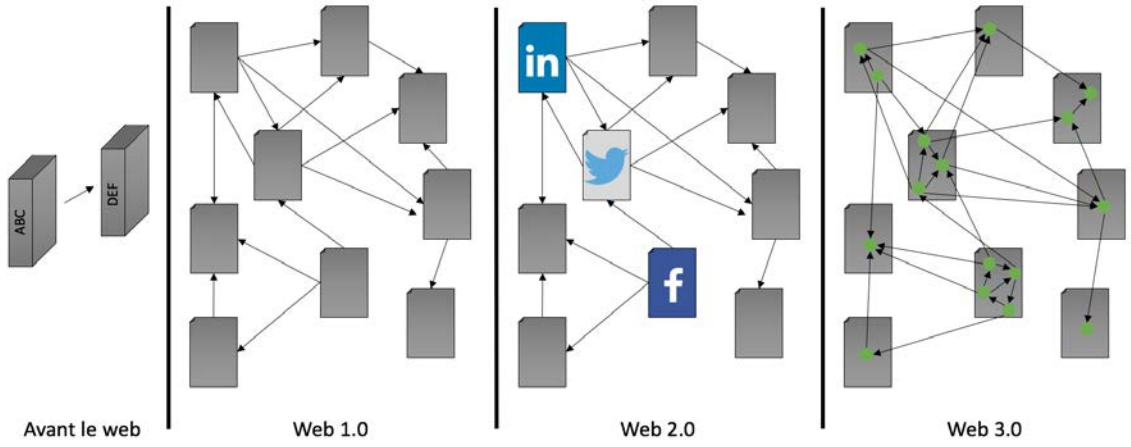

Figure 2.7 – Evolution du web jusqu'au web sémantique

Le web sémantique voudrait tirer profit de l'énorme quantité de données et pouvoir manipuler cette connaissance. Dans ce but, l'identification des ressources et des informations est cruciale. Cela nécessite une description des documents et des ressources [38]. Les standards présentés dans la section antérieure et les ontologies ont clairement été conçus pour atteindre cet objectif. Le web sémantique a permis d'ouvrir les portes à une représentation formelle de la connaissance, et qui plus est, compréhensible par une machine. Alors que les ontologies ont au départ été créées pour le web sémantique, elles peuvent évidemment être utilisées en dehors du web [34].

2.2 Logiciels Concernés

Cette section concerne les logiciels qui traitent des ontologies et qui ont été étudiés durant le projet IFLEC. Les sous-sections suivantes présentent de manière très succincte les logiciels.

2.2.1 OLED

Le premier logiciel étudié est OLED (*OntoUML Lightweight Editor*) [39]. OLED est en réalité une version autonome du logiciel OntoUML. Alors que ce dernier est à utiliser avec une application tierce, OLED permet de travailler directement sur un seul logiciel. La fonctionnalité principale du programme est la création d'ontologies de manière graphique. Il suffit de déposer des formes et des liens pour créer l'ontologie du domaine souhaité. Par ailleurs, il s'agit d'une extension de UML qui permet de créer des ontologies. Une fonctionnalité permet l'exportation directement vers le langage OWL.

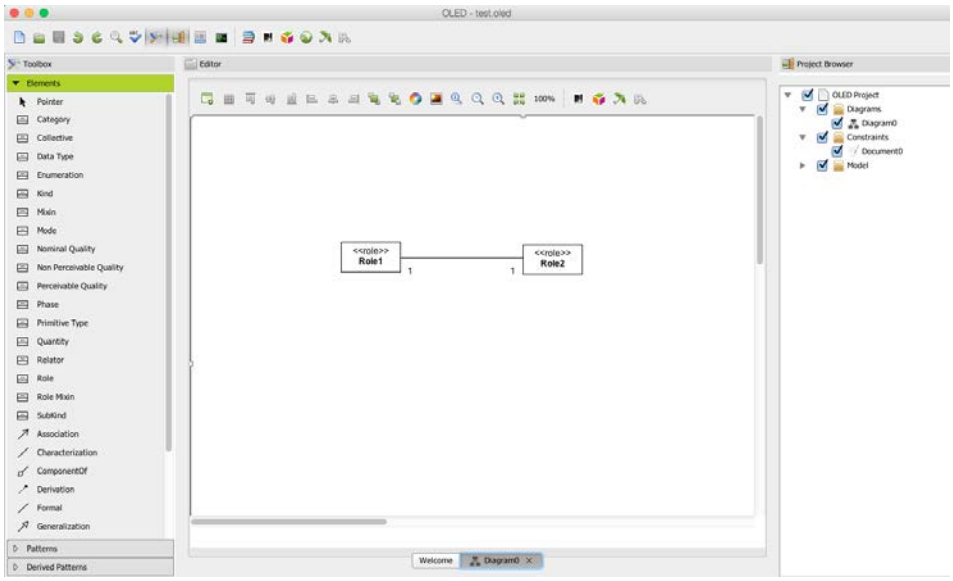

Figure 2.8 – Capture d'écran du logiciel OLED

2.2.2 Jena

Jena [40] n'est pas un logiciel mais une API (*Application Programming Interface*) de création d'ontologies qui doit être vue comme bibliothèque de fonctions et de méthodes. Cette API fonctionne sous Java et peut donc être intégrée dans un environnent de développement Java comme Eclipse [41]. L'utilisation de Jena nécessite l'utilisation de lignes de codes pour construire une ontologie. L'API supporte les langages RDF et OWL. Elle propose aussi des outils pour manipuler les données et raisonner sur l'ontologie créée à l'aide de moteurs d'inférence [40]. Les moteurs d'inférence permettent d'inférer sur un ensemble de règles et ajouter de l'information supplémentaire à celle déjà existante [42].

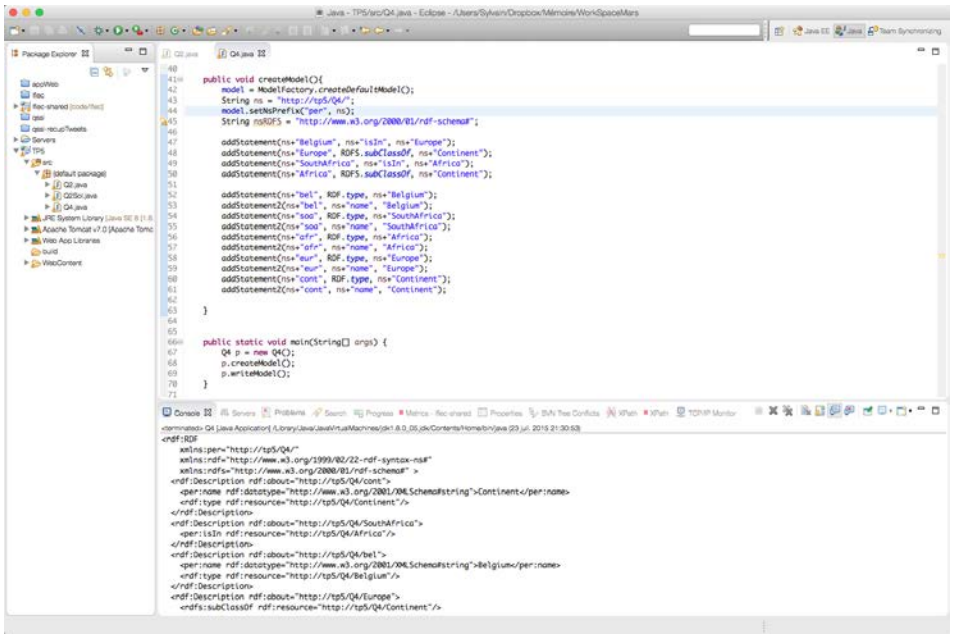

Figure 2.9 – Capture d'écran de l'utilisation de Jena dans Eclipse

2.2.3 Protégé

Le logiciel Protégé [43] est considéré par beaucoup d'experts comme le meilleur outil de conception d'ontologies. Il a été créé à l'Université de Standford par une équipe en recherche médicale. Ce logiciel peut gérer de nombreux formats d'ontologies dont ceux présentés précédemment dans ce chapitre. Il a été reconnu pour sa capacité à gérer la complexité d'ontologies de grandes dimensions. Protégé possède également ses moteurs d'inférence pour raisonner sur les ontologies. Il faut noter qu'en plus de la version de bureau, une version plus légère a vu le jour sur le web sous le nom de WebProtégé [43].

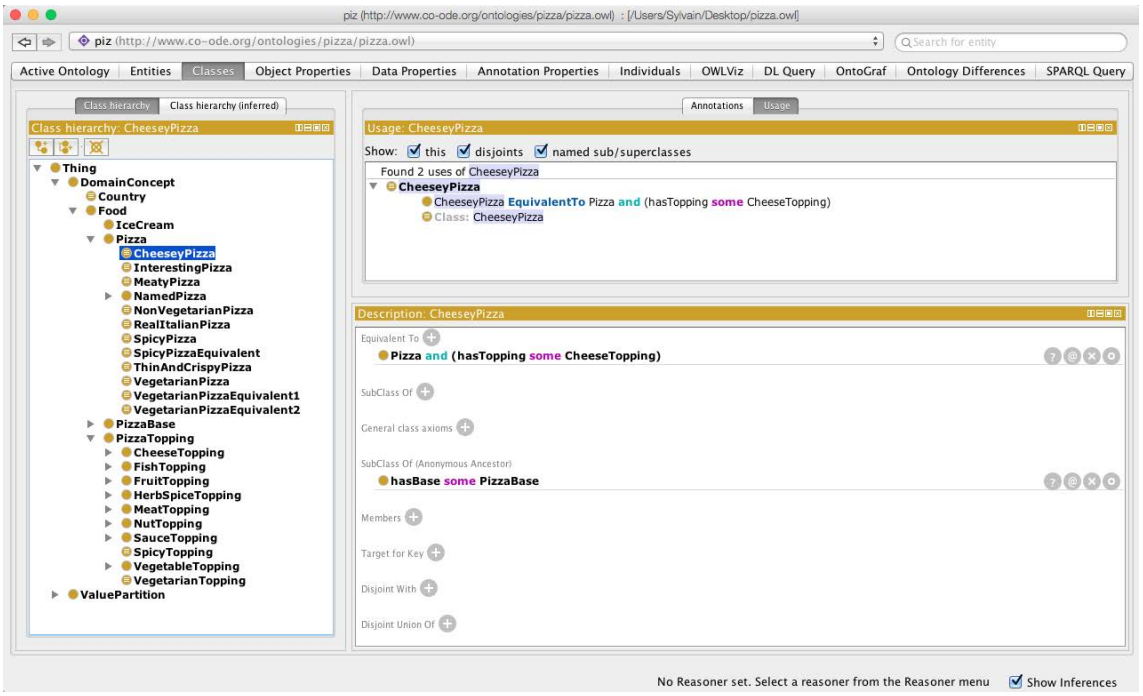

Figure 2.10 – Capture d'écran de Protégé

2.2.4 NeOn

NeOn [44] est un projet de recherche européen qui a débuté en 2006 pour se terminer en 2010. Durant ces quatre années, le projet a impliqué pas moins de 14 pays européens pour le développement d'un outil de création et de visualisation d'ontologies. Le but des chercheurs était de faire avancer la recherche dans le domaine du web sémantique. NeOn est basé sur une plateforme Eclipse et permet, avec une certaine facilité, la construction d'ontologies. L'outil inclut une partie création d'ontologies de façon intuitive par des options *Ajouter une classe* ou *Ajouter un individu*. La seconde partie du logiciel permet de parcourir les éléments, construits au préalable, via un outil de visualisation dynamique [44].

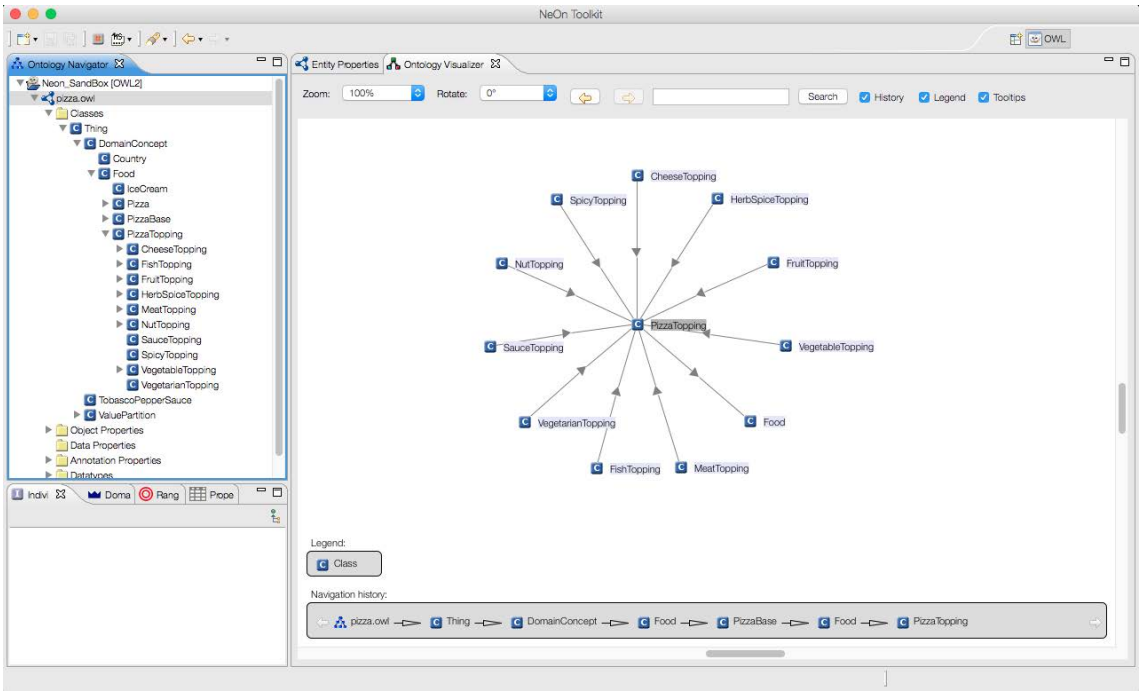

Figure 2.11 – Capture d'écran de NeOn

2.2.5 D2RQ

La plateforme D2RQ [45] est uniquement liée aux ontologies par son utilisation. Cette plateforme constitue un système pour parcourir une base de données relationnelle en tant que graphe RDF virtuel. Les données sont contenues dans la base de données mais le système se compare à du web sémantique. Il n'y a donc aucune duplication des données. Pour obtenir une base de données basée sur le web sémantique, des fichiers de mapping sont nécessaires. Ils autorisent le couplage entre les éléments de la base de données et une ontologie créée au préalable [45].

La plateforme D2RQ se compose de trois éléments :

1. **Un langage de couplage :** les liaisons entre les données et l'ontologie se font via l'écriture d'un fichier de mapping. Ce document est en réalité un document RDF.

2. **Le moteur D2RQ :** il s'agit d'un plug-in pour Jena qui utilise les fichiers de mapping pour réaliser les requêtes sur la base de données relationnelle de départ.

3. **Le serveur D2R :** le serveur D2R est un serveur HTPP qui fournit une interface sommaire pour parcourir les données transformées suivant le web sémantique.

La figure 2.12 illustre le fonctionnement de D2RQ.

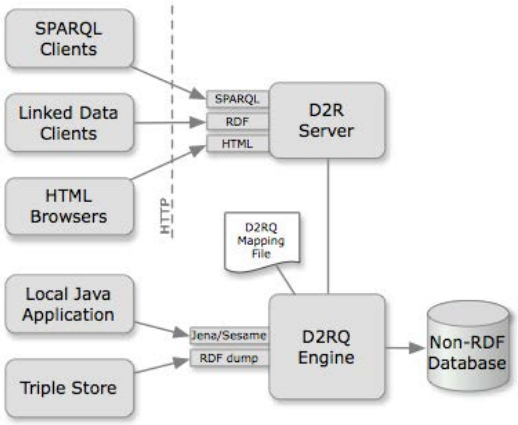

Figure 2.12 – Architecture complète de D2RQ [45]

2.3 Importance dans le Milieu Médical

Les données des patients et toutes les informations concernant leurs traitements dans les hôpitaux proviennent souvent de sources hétérogènes. Leur format n'est parfois pas adapté pour de l'analyse (données textuelles, etc.) et nécessite d'être organisées et liées à des ressources externes de connaissances [46]. Certains groupes de travail ont pris conscience de cette problématique et tentent de la résoudre en développant de nouveaux logiciels. L'aide à la décision mais aussi le traitement automatique des données peut être rendu possible grâce à des méthodes améliorées basées sur des ontologies. L'information contenue dans les bases de données est alors adaptée par des méthodes d'inférence [47].

Pour pouvoir être efficaces, les systèmes hospitaliers devraient pouvoir utiliser un modèle de représentation unique pour un même concept du domaine. Les échanges entre établissements et les évolutions des systèmes ne favorisent pas cette représentation. Deux solutions s'offrent à eux : réaliser une transformation (soit des données reçues, soit faire évoluer ses propres données pour respecter un nouveau modèle) ou alors faire une utilisation massive d'ontologies pour lier les données aux connaissances médicales. La deuxième manière de faire est étudiée dans de nombreux hôpitaux. L'intégration des sources multiples est favorisée par l'utilisation d'une sémantique partagée offerte par des ontologies, des terminologies ou des thésaurus [46].

A travers ce chapitre, il a été dit qu'une ontologie était utilisée comme supplément d'informations et était devenue indispensable à l'exploitation des données. Les terminologies détaillent, tout comme les ontologies un domaine précis, mais ne le représentent que par des listes de termes structurées ou non. Les thésaurus représentent les vocabulaires d'un domaine particulier mais organisés formellement. Les termes sont organisés entre eux par des relations sémantiques comme des relations hiérarchiques, associatives ou d'équivalence [30]. Coupler les trois éléments avec les données des hôpitaux n'est pas inutile et pourrait grandement augmenter la capacité, pour une machine, de réaliser des traitements automatiques. La figure 2.13 montre la classification des ontologies en fonction du spectre sémantique (adapté de [30]).

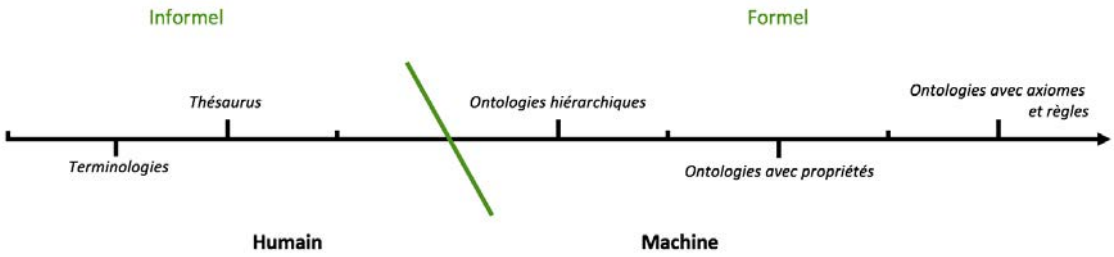

Figure 2.13 – Classification des ontologies en fonction du spectre sémantique

Avec tout ce qui a été expliqué à travers ce chapitre, une question se pose :

Pourquoi faut-il choisir les ontologies et les outils du web sémantique dans les systèmes informatiques tels que ceux utilisés dans les hôpitaux ?

Pour y répondre, il faut montrer que l'exploitation classique des bases de données cliniques ne permet pas de traiter des questions médicales qui nécessitent une connaissance spécifique d'un domaine [46]. Ainsi, une illustration par l'exemple [4] servira de fil conducteur tout au long de l'explication.

Soit une base de données contenant plusieurs tables dont celles-ci :

— **bacterie :** contient les noms des bactéries ;
— **antibiotique :** contient les noms des antibiotiques ;
— **antibiogramme :** contient les résultats de sensibilité et de résistance d'une souche bactérienne envers certains antibiotiques.

Soit une ontologie conçue expressément pour la base de données utilisée qui décrit le domaine des maladies infectieuses comme les agents pathogènes, les prescriptions médicamenteuses ou encore les diagnostics. Pour veiller à standardiser les termes contenus dans la base de données, des correspondances ont été créées entre les tables et des terminologies externes (dictionnaire médical SNOMED CT [48])

Soit l'utilisation du serveur D2R pour générer le graphe virtuel RDF de la base de données automatiquement. Ce fichier est conforme à la notation en triplets vue dans la section précédente.

Soit un exemple de requête faite par un médecin : *Quels sont les résultats d'antibiothérapie d'une infection d'E. Coli, résistante aux fluoroquinolones ?*

Dans un système informatique hospitalier classique, les requêtes faisant intervenir un élément particulier, comme un antibiotique, émettent les résultats qui les concernent et rien de plus. La requête précédente ne proposera que les résultats liés à la *fluoroquinolone*. Imaginez le cas où la donnée est inexistante !

4. L'exemple s'inspire de l'article sur le web sémantique dans le milieu hospitalier [46]

Dans le cas d'une base de données basée sur le web sémantique, faisant intervenir la base de données relationnelle, les fichiers d'ontologies et le serveur D2R, les résultats seront différents. Suivant l'organisation des éléments dans l'ontologie, des résultats supplémentaires peuvent venir s'ajouter. L'ontologie utilisée pour l'exemple a défini une organisation hiérarchique des antibiotiques en créant des sous-classes. De ce fait, l'*Ofloxacine* est une *Quinolone*. Cette dernière est aussi considérée comme étant équivalente à la *Fluoroquinolone*. Le résultat de la même requête inclura donc l'*Ofloxacine* car elle est comprise comme une *Fluoroquinolone*.

La figure 2.14 réalise un résumé synthétique du processus d'association d'une base de données à une ontologie, comme dans l'exemple précédent [5].

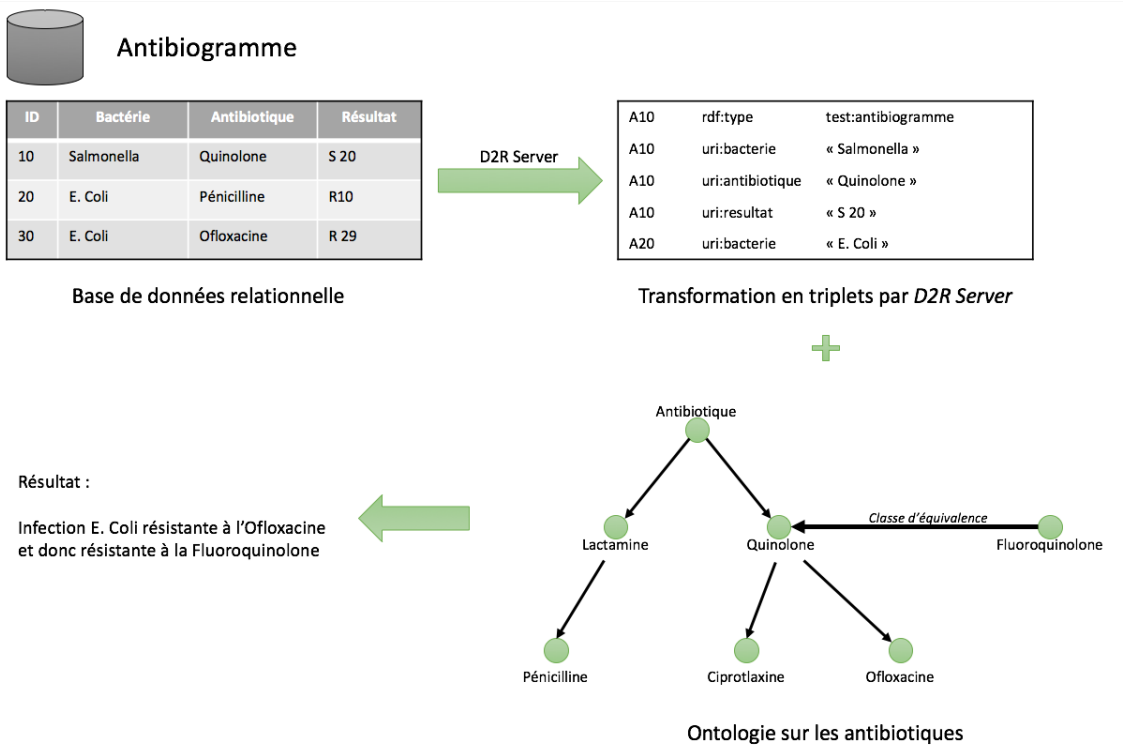

Figure 2.14 – Processus d'association des données

Les technologies du web sémantique permettent d'obtenir des résultats incluant les connaissances de différents domaines comme pour les antibiotiques. Les ontologies viennent comme support aux systèmes mais surtout comme données [46]. A l'heure actuelle, elles sont considérées comme un passage obligé par la communauté informatique médicale [49].

Afin de bénéficier de ce genre de système, basé sur le web sémantique, des adaptations sont à effectuer. Il y a trois cas à envisager [46] :

5. Le « S » et le « R » dans la colonne « Résultat » signifie *sensible* et *résistant* suivant une valeur déterminée.

1. Utiliser les **entrepôts RDF** qui stockent les données sous forme de triplets RDF dans de simples fichiers.

2. Utiliser les **base de données à base ontologique** où les données et les ontologies sont stockées au même endroit. Les ontologies se comportent comme les données et les opérations d'insertion, de modification et de suppression sont autorisées. Un exemple de ce type de base de données est OntoDB. Un nouveau langage d'interrogation est nécessaire : Express.

3. Utiliser le **SPARQL Endpoint** qui vise à aligner le modèle d'information sur l'ontologie. Il s'agit de l'approche illustrée par l'exemple précédent. La base de données est considérée comme un graphe virtuel. L'avantage est de garder la base de données telle quelle, sans réel changement dans sa conception. Il faut par contre ajouter d'autres systèmes comme le serveur D2R et les fichiers d'ontologies. Le langage d'interrogation change et passe du SQL au SPARQL.

Chapitre 3

Data Mining

Ce chapitre sur le data mining (ou *exploration de données* en français) fait la synthèse des concepts et des méthodes utilisées pour traiter les gigantesques bases de données qu'utilisent les plus grandes entreprises du monde mais pas seulement. Les établissements de soins voient leur quantité de données augmenter au fil du temps et la question pour les traiter efficacement est posée par les experts. Les sections de ce chapitre visent à faire le point sur le sujet.

3.1 Présentation du Data Mining

Les sous-sections suivantes tentent d'expliquer ce qu'est le data mining et quelles sont les techniques à utiliser.

3.1.1 Évolution vers l'Ère de l'Information

Le monde d'aujourd'hui est caractérisé par le nombre toujours grandissant de données récoltées. Rien que pour le réseau social Facebook, environ 600 To [1] de données sont échangées chaque jour. Cela équivaut à pas moins de 750.000 CD, 130.000 DVD ou encore 22.000 disques Blu-ray. Les serveurs de Facebook stockent aujourd'hui une quantité totale de 300 Po [2] [50]. Le réseau social n'est pas le seul à posséder de grandes quantités de données.

1. 1 To ou 1 Téraoctet est une unité de mesure correspondant à 1.000 Go (Gigaoctets)
2. 1 Po ou 1 Pétaoctet correspond à 1.000 To (Téraoctet)

Figure 3.1 – Problématique de posséder de l'information [51]

En effet, la société actuelle a réussi à faire vivre les Hommes dans l'ère du numérique et de l'information. Et quelle information ! Que ce soient les entreprises, les réseaux d'ordinateurs, les sciences et l'ingénierie, ou encore la médecine, quasi tous les aspects de la vie quotidienne produisent des données virtuelles. L'explosion du volume de données est le résultat de l'informatisation de la société et aucun domaine n'est épargné. La croissance de la quantité de données mais aussi de sa disponibilité a fait passer la société à l'ère des données [51].

Les données sont présentes en nombre dans les bases de données mais l'information disponible ne l'est pas forcément. En effet, de puissants outils sont plus que nécessaires pour dévoiler l'information utile parmi les quantités astronomiques de données brutes (voir figure 3.1). Cette nécessité de posséder une connaissance organisée a donné naissance au data mining [51].

3.1.2 Concept de Data Mining

Le data mining est un terme anglais pour désigner l'extraction de connaissance à travers de grandes quantités de données. Le terme en lui-même est inapproprié car il désigne littéralement l'extraction de données. Or, ce n'est pas le cas et *knowledge mining*[3] aurait été plus adapté [51]. Par extraction de connaissance, la discipline entend trouver des relations inattendues et résumer les abondants ensembles de données [52].

3. Le terme signifie littéralement *extraction de connaissance*. Il aurait été plus juste de l'utiliser plutôt que *data mining*.

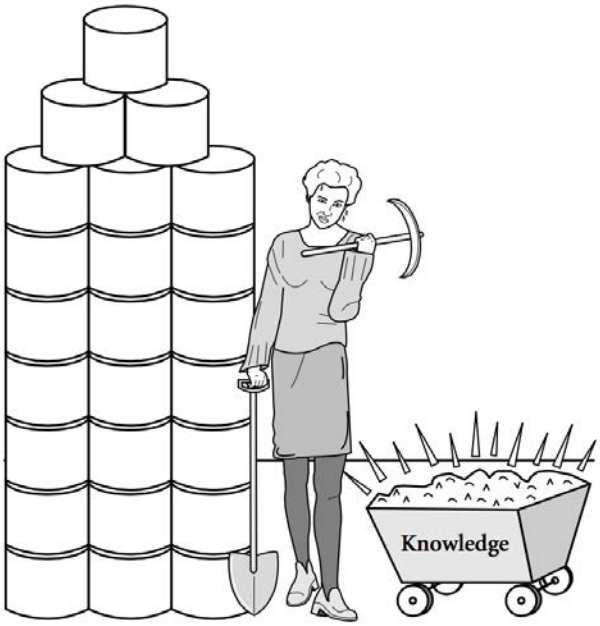

Figure 3.2 – Recherche de la connaissance à travers les données [51]

Le data mining n'est pas à confondre avec le KDD (*Knowledge Discovery from Data* - Extraction de connaissance à partir de données) qui est le processus complet permettant l'extraction de la connaissance. Le data mining est inclus dans le KDD car il en est l'étape essentielle. Le processus complet du KDD est représenté à la figure 3.3. Ce processus est composé de plusieurs étapes [51, 52] :

1. **Nettoyage des données :** la première étape consiste à supprimer les données inconsistantes et les doublons.

2. **Intégration des données :** dans le cas où les sources sont multiples, il faut au préalable combiner les bases de données entre elles.

3. **Sélection des données :** afin de réaliser une analyse correcte, il est nécessaire de supprimer les données qui n'apporteraient rien dans la connaissance qu'on vise à extraire

4. **Transformation des données :** la quatrième phase est une phase de transformation des données suivant un format accepté par les logiciels de data mining. Il s'agit d'opérations d'agrégation, de synthèse et de mise en forme.

5. **Data mining :** l'étape essentielle est réalisée sur des données clés par des méthodes intelligentes d'extraction.

6. **Interprétation :** cette dernière étape consiste à évaluer et identifier les modèles de représentation de la connaissance qui apparaissent mais aussi représenter graphiquement la nouvelle information.

Les différentes étapes qui ont été identifiées peuvent varier suivant les experts. L'une ou l'autre étape peut être réalisée avant une autre jusqu'à l'étape de data mining. Ce groupe de 4 phases est appelé *préprocessing* et vise à préparer les données avant leur exploration. Leur ordre n'est pas figé mais celui présenté est le plus cohérent [51].

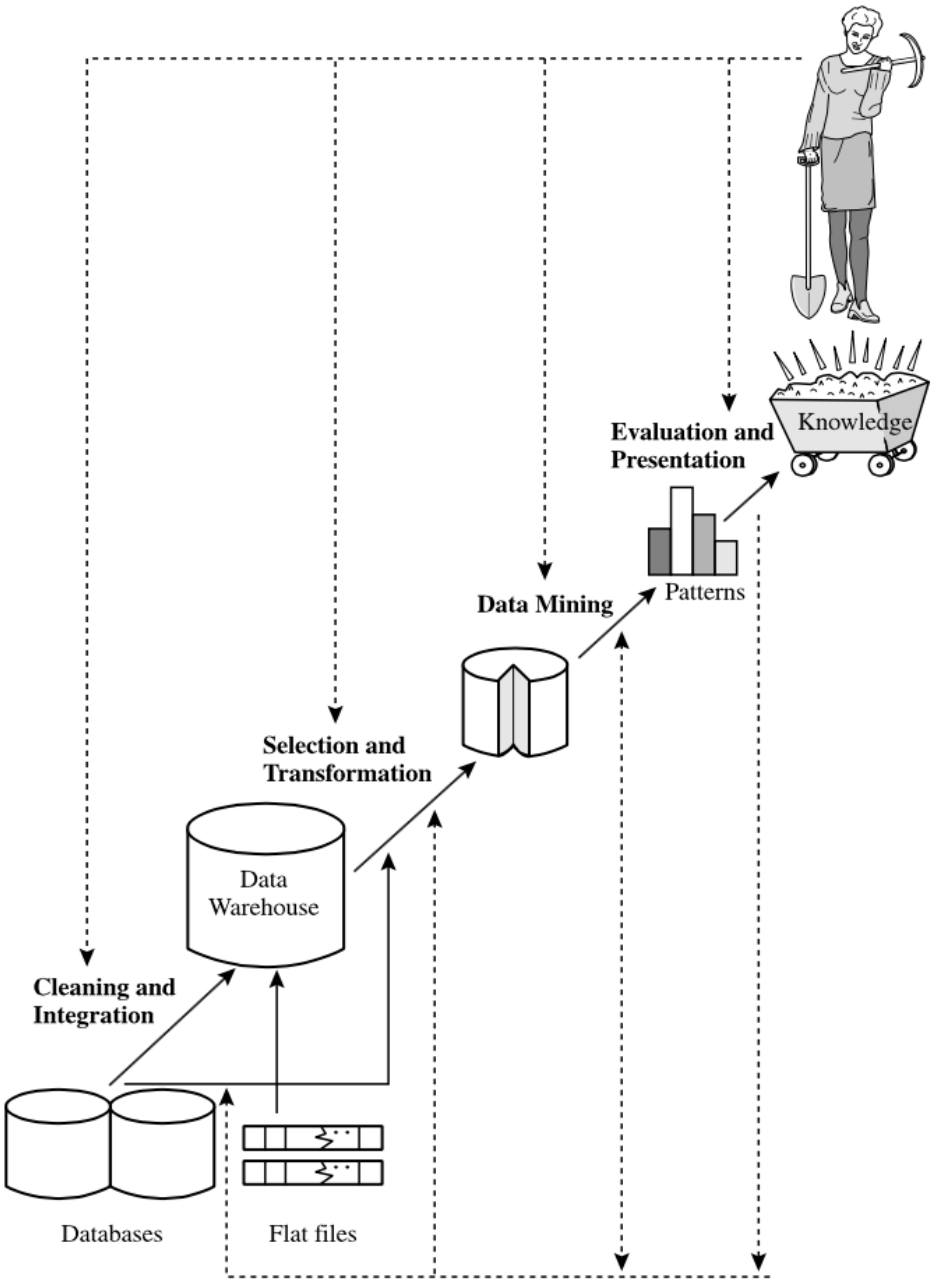

Figure 3.3 – Illustration des étapes du KDD [51]

Le data mining représente la phase la plus importante du KDD et ne constitue pas à lui seul le processus complet pour rechercher la connaissance. Il est plutôt défini comme une façon de trouver des relations cachées entre les données [52] ou une technique utilisée pour découvrir de nouveaux regroupements de données à partir de celles contenues dans grandes bases de données [53]. Le data mining permet de synthétiser des données mais également de rendre des résumés statistiques [54].

3.1.3 Traitement des Données

Pour traiter l'énorme quantité de données, le data mining peut lui-même être séparé en plusieurs phases à l'instar du KDD. Ces phases n'apparaissent pas comme telles dans la littérature mais en sont déduites. Ainsi, trois étapes sont présentes au sein du data mining :

1. **Choisir les méthodes d'exploration.** Parmi un large choix d'algorithmes, il est possible de sélectionner celui qui convient le mieux à la situation de départ.

2. **Explorer les données.** Durant cette deuxième étape, la méthode ou l'algorithme sélectionné par l'étape précédente est exécuté.

3. **Générer les modèles.** La méthode d'exploration produit une série de résultats et des modèles (patterns) de données sont générés. Un système de data mining génère des milliers de modèles mais ils ne sont pas tous intéressants. Durant la dernière phase du KDD, un traitement manuel est requis pour vérifier l'intérêt d'un modèle. Un modèle est intéressant [54] s'il est :

   — facilement compris par les humains ;
   — valide sur des données nouvelles ;
   — potentiellement utile ;
   — nouveau, ou validant certaines hypothèses.

La figure 3.4 représente les phases incluses dans le data mining qui sont au nombre de trois. Pour choisir la méthode d'exploration appropriée, il est conseillé de lire la section 3.2 sur les techniques d'exploration.

Figure 3.4 – Illustration des différentes phases du data mining

Le traitement sur les données par le data mining est très différent des requêtes traditionnelles sur les bases de données. Ces dernières sont des requêtes précises sur les données à l'aide du langage SQL généralement. Le data mining n'a parfois pas d'objectif précis avant de lancer une méthode ou un algorithme d'exploration de données. Aussi, au contraire des requêtes SQL, les résultats en sortie de data mining n'est pas un sous-ensemble des données [55]. Il s'agit de nouvelles données calculées sur base des anciennes.

Le data mining n'est pas une nouvelle approche pour traiter les données et de nombreuses techniques provenant de domaines différents s'y retrouvent intégrées. La nature interdisciplinaire du data mining contribue à son succès et à ses applications pratiques. Pour ne citer qu'eux, les domaines dont certaines méthodes et algorithmes ont été inclus sont les statistiques, le machine learning, l'algorithmique, les sciences de l'information, la recherche d'information, les bases de données et la reconnaissance de modèles [51]. D'autres disciplines ont également pu influencer le data mining mais ne sont pas reprises ici. La figure 3.5 illustre l'influence des différentes disciplines sur l'exploration de données.

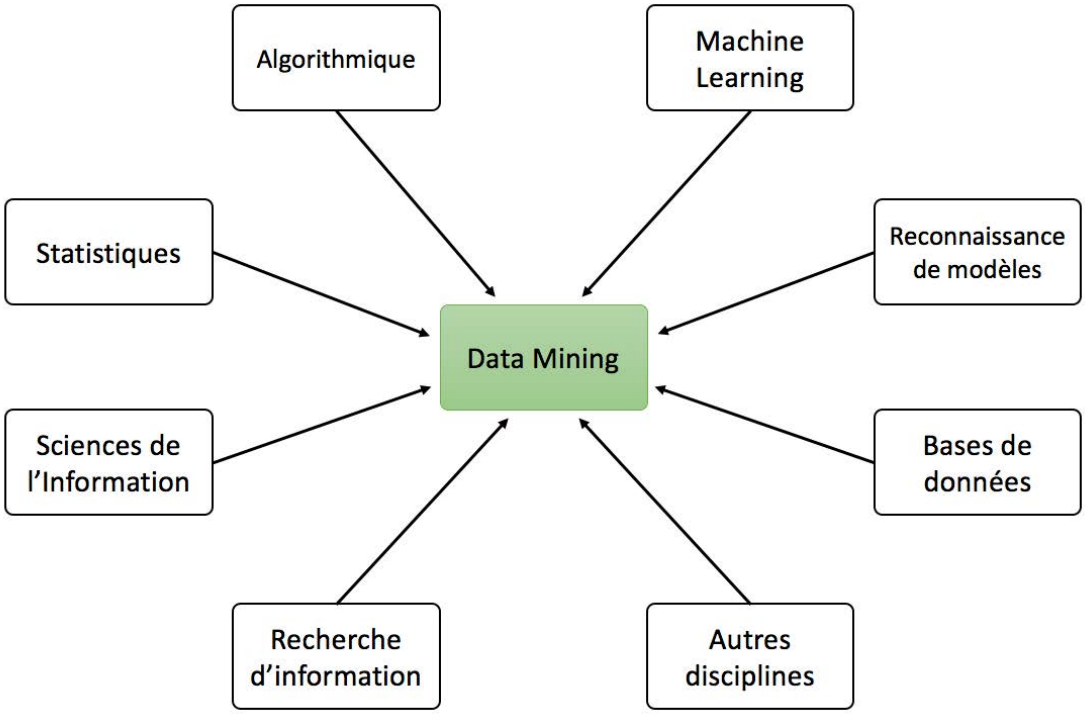

Figure 3.5 – Illustration de l'influence de nombreux domaines sur le data mining

3.2 Techniques d'Exploration

Avant de détailler les algorithmes et les méthodes qui composent le data mining, il est nécessaire de savoir que le point de départ de tout processus de data mining est l'analyse exploratoire des données. Elle permet de donner des statistiques sommaires sur les données mais également des représentations graphiques. Par là, l'utilisateur peut visualiser les données sous forme schématique [55].

Cette section donne donc une vision générale des techniques utilisées par le data mining. Celles-ci se décomposent en deux groupes et sont appelées techniques prédictives ou techniques descriptives. Elles englobent plusieurs catégories de méthodes et d'algorithmes. La figure 3.6 fait la synthèse des éléments.

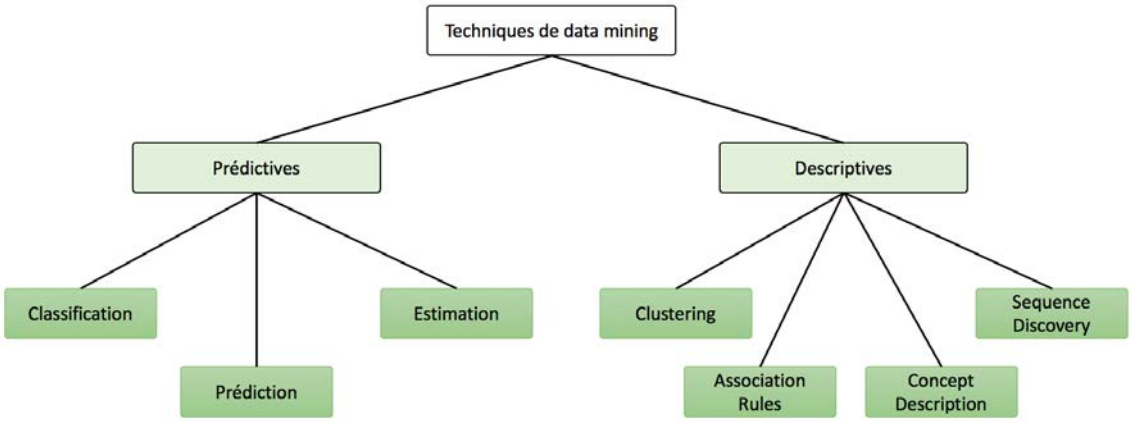

Figure 3.6 – Classification des techniques utilisées par le data mining

3.2.1 Techniques Prédictives

Le premier regroupement d'algorithmes a pour nom « techniques prédictives » ou « techniques supervisées ». Comme leur nom l'indique, les techniques prédictives visent à réaliser des prédictions d'évolution à partir des données explorées. Dans cette catégorie, plusieurs groupes sont mis en évidence, contenant chacun un ou plusieurs algorithmes. Les techniques prédictives comportent quatre classes d'algorithmes et de méthodes. Celles-ci sont détaillées dans la suite.

Classification La classification consiste à prédire des valeurs sur base de données en entrée. Pour ce faire, les données sont classées dans des groupes prédéfinis (classes ou catégories) [55, 56]. Un tel classement est possible grâce à l'implémentation particulière des algorithmes. Pour prédire les valeurs futures, les algorithmes s'entraînent sur des données spéciales. Il s'agit généralement d'un ensemble de données appelé « ensemble d'entraînement » (*training set*). L'entraînement tente de découvrir les relations qui peuvent exister entre attributs. Suivant les résultats de l'entraînement, les valeurs finales pourront être générées pour les entrées déterminées qui se trouvent en dehors de l'ensemble d'entraînement. Ce nouvel ensemble est appelé « ensemble de prédiction » (*prediction set*) [56].

Ensemble d'entraînement

Âge	Pulsation	Pression sanguine	Problème cardiaque
65	78	150/70	oui
37	83	112/76	non
71	67	108/65	non

Ensemble de prédiction

Âge	Pulsation	Pression sanguine	Problème cardiaque
43	98	147/89	?
65	58	106/63	?
84	77	150/65	?

Figure 3.7 – Exemple d'ensembles d'entraînement et de prédiction

L'exemple à la figure 3.7 (adapté de [56]) illustre l'entraînement sur le premier ensemble de données et les valeurs à prédire pour la suite. Ainsi, l'algorithme s'entraîne un certain nombre de fois sur les données dont il dispose. Il peut ensuite prédire, dans le cas de cet exemple, si un problème cardiaque est lié à l'âge, à la pulsation du cœur ou à la pression sanguine.

La représentation de la connaissance dans la classification est la création de règles de prédiction qui se présentent sous la forme de règles SI-ALORS [56]. Une telle forme possède deux côtés : les prémisses (partie SI) et la conclusion (partie ALORS). Les prémisses sont les conditions à respecter pour vérifier la conclusion. Pour continuer l'exemple sur les problèmes cardiaques, une règle de prédiction pour la première ligne de l'ensemble d'entraînement pourrait être :

```
SI (âge=65 ET pulsation>70) OU (âge>60 ET pression sanguine>140/70) ALORS problème cardiaque=OUI
```

Dans la réalité, les règles sont beaucoup plus complexes et beaucoup plus nombreuses. Avec l'ensemble de ces règles, la classification définit des classes, des catégories de données [56]. L'intérêt de cette méthode de data mining réside dans l'estimation de valeurs inconnues [57], comme ici, estimer le risque de problèmes cardiaques.

Au sein de cette catégorie de data mining, plusieurs algorithmes ont trouvé leur place. La liste non exhaustive présente ci-dessous les différents types d'algorithmes possibles.

— **Réseaux de neurones**
— **Techniques statistiques** (k plus proches voisins, ...)
— **Arbres de décision** (arbres aléatoires, forêts aléatoires, J48, ...)

Estimation Au contraire de la classification, l'estimation consiste à définir le lien entre un ensemble défini de données et une variable cible. Dans la classification, l'algorithme travaille sur une variable cible catégorielle [4] alors que l'estimation travaille sur une variable cible numérique. L'estimation crée de nouvelles valeurs pour une variable cible inconnue. Ceci veut dire que toutes les valeurs sur lesquelles s'exécute l'algorithme sont définies (tant les prémisses que la conclusion). L'estimation est utilisée alors pour prédire une valeur autre que celles manipulées [57].

Par exemple, l'estimation est utilisée pour estimer la pression sanguine à partir de l'âge du patient, son sexe, son poids et son niveau de sodium dans le sang. Son intérêt est aussi de pouvoir prévoir des valeurs inconnues par le système. Les algorithmes qui peuvent être utilisés sont les suivants :

— **Analyse statistique** (régression linéaire simple, corrélation, régression multiple, intervalle de confiance, ...)
— **Réseaux de neurones**

Prédiction Alors que la classification propose un classement de valeurs discrètes, la prédiction fonctionne suivant des modèles à valeurs continues. La prédiction est surtout utilisée pour prédire des données numériques, qui sont manquantes ou indisponibles, sans créer de classement. Bien que d'autres méthodes existent aussi, *l'analyse de régression* est une méthode statistique qui est le plus souvent utilisé pour la prévision numérique. La prédiction englobe également l'identification des tendances de distribution sur la base des données disponibles [51]. La prédiction est tout de même similaire à la classification et à l'estimation mise à part que pour la prédiction, les résultats portent sur le futur et non le présent [57].

L'intérêt de la prédiction est de permettre la prévision de valeurs inconnues par des algorithmes similaires à la classification et à l'estimation. Ce genre d'algorithmes peut servir à prévoir le prix d'actions à trois mois ou prévoir le temps qu'il fera dans plus d'une semaine [57].

— **Techniques statistiques** (k plus proches voisins, ...)
— **Régression**
— **Réseaux Bayésiens**

3.2.2 Techniques Descriptives

Le deuxième regroupement d'algorithmes a pour nom « techniques descriptives » ou « techniques non supervisées ». Comme leur nom l'indique, les techniques descriptives visent à décrire les données explorées en mettant en évidence les informations présentes mais cachées par le volume de données. Le but est donc de résumer, synthétiser, classer plutôt que d'extrapoler de nouvelles informations. Tout comme dans le cas des techniques prédictives, les techniques descriptives comportent plusieurs catégories d'algorithmes qui sont détaillées dans la suite.

4. Une variable catégorielle est une variable qui permet de faire des regroupements de valeurs par catégorie. Il s'agit de classements. Elle est opposée à une variable numérique qui permet, elle, de faire des synthèses sur les données comme une moyenne, un minimum, un écart-type, etc.

Clustering Premier type d'algorithmes faisant partie des techniques descriptives, le clustering vise à regrouper un ensemble de données en différents groupes homogènes. Chaque groupe englobe les éléments qui ont des caractéristiques communes et qui correspondent à des critères de proximité. Dit autrement, il s'agit de générer des sous-ensembles disjoints de données. Ces sous-ensembles sont appelés *clusters* [58]. Le clustering est semblable à la classification à ceci près que les classes ne sont pas définies à l'avance mais déterminées par l'algorithme lui-même lors de son exécution [55].

Pour créer des clusters, les algorithmes veillent à minimiser la distance intra-classe (élément homogène au sein de la classe) et à maximiser la distance extra-classe pour obtenir des ensembles le plus distinct possible [58]. Les algorithmes qui existent sont les suivants :

— **Classification hiérarchique**
— **Classification des k moyennes**
— **Réseaux de Kohonen**

Association Rules Un autre type d'algorithmes est celui des règles d'association aussi appelé *Association Rules*. Les règles d'association sont liées aux bases de données relationnelles qui décrivent des liaisons, des associations entre les éléments. Cette technique de data mining y est similaire dans le sens où elle fait ressortir les liens entre les données. L'origine des règles d'association vient du marketing où les supermarchés voulaient analyser les données de leurs ventes [58].

Pour trouver quelles valeurs vont ensemble, l'algorithme réalise une analyse d'affinité et crée des règles d'association entre les éléments. Par exemple, une analyse du panier de la ménagère conduirait à des règles comme celles-ci [55, 58] :

— SI un client achète des fraises, ALORS il achète des cerises.
— SI un client achète du pain et du beurre, ALORS il achète 9 fois sur 10 du lait en même temps.

Utilisé pour mieux connaître les comportements, différents algorithmes permettent d'utiliser les règles d'association. Ceux-ci sont par exemple :

— **Apriori**
— **Partition**
— **Sampling**
— **Eclat**

Concept Description Comme son nom l'indique le *Concept Description* ou description de concepts, consiste à décrire un concept représenté par un ensemble de données. Cette description tient à présenter la répartition des valeurs par rapport aux autres en générant des statistiques (moyenne, minimum, maximum, etc.) ou des graphiques. L'intérêt d'une telle technique est de favoriser la connaissance et la compréhension des données [57].

Un exemple concret serait de calculer, lors d'une élection, la répartition des votes par tranche d'âge pour marquer le lien entre les variables « vote » et « âge ». Pour ce faire, ce sont essentiellement des méthodes graphiques qui sont utilisées [57].

Sequence Discovery La dernière catégorie d'algorithmes pour les techniques descriptives concerne le *Sequence Discovery*. Littéralement *recherche de séquences*, cette technique concerne la détection de modèles (patterns) dans de longs flux de données dont les valeurs sont délivrées par séquences. La recherche de séquences est très fortement liée à la biologie, pour le séquencement de l'ADN ainsi que pour l'analyse des gènes et des protéines. L'analyse de textes utilise également cette technique car les phrases sont considérées comme des séquences ordonnées de mots [58].

Concernant le domaine de la biologie, le *Sequence Discovery* n'est pas utilisé pour connaître la séquence des nucléotides dans l'ADN mais bien pour comprendre la séquence et en définir sa structure.

3.2.3 Synthèse des Éléments

Les **techniques descriptives** sont destinées à décrire, résumer, synthétiser et classer les données. Elles sont utilisées pour mettre en évidence des informations qui sont présentes dans l'ensemble des données collectées mais qui y sont cachées. Elles ne créent pas de nouvelles données.

Les quatre types de méthodes pour les techniques descriptives sont :

— le clustering
— les règles d'association
— la description de concepts
— la recherche de séquences

Les **techniques prédictives** sont utilisées pour prédire ou extrapoler de nouvelles informations à partir des données récoltées. De nouvelles données sont donc créées pour une variable choisie au préalable. L'objectif des techniques prédictives est de prédire les valeurs de la variable qui a été choisie, mais aussi de réaliser un classement à partir de cette variable. Les algorithmes de cette catégorie de techniques sont beaucoup plus difficiles à mettre en place.

Les trois types de méthodes pour les techniques prédictives sont :

— la classification
— l'estimation
— la prédiction

3.3 Outils d'Exploration de Données

La présente section traite des outils d'exploration de données, c'est-à-dire des logiciels qui proposent les algorithmes de data mining cités dans la section précédente. De nombreux logiciels gratuits ou payants existent, offrant des outils complets ou moins complets selon les cas. Deux d'entre eux, gratuits, sont présentés ici : Weka et R.

Weka Le premier élément sur lequel il faut se concentrer est Weka (*Waikato Environment for Knowledge Analysis*) qui a été développé à l'Université de Waikato en Nouvelle-Zélande [59]. Il a ceci de particulier : il peut être utilisé en tant que logiciel ou en tant que libraire. Cette dernière autorise un programmeur à employer les méthodes incluses dans Weka directement dans l'application qu'il veut créer. L'outil a entièrement été développé en Java et est donc compatible avec quasiment toutes les plateformes modernes.

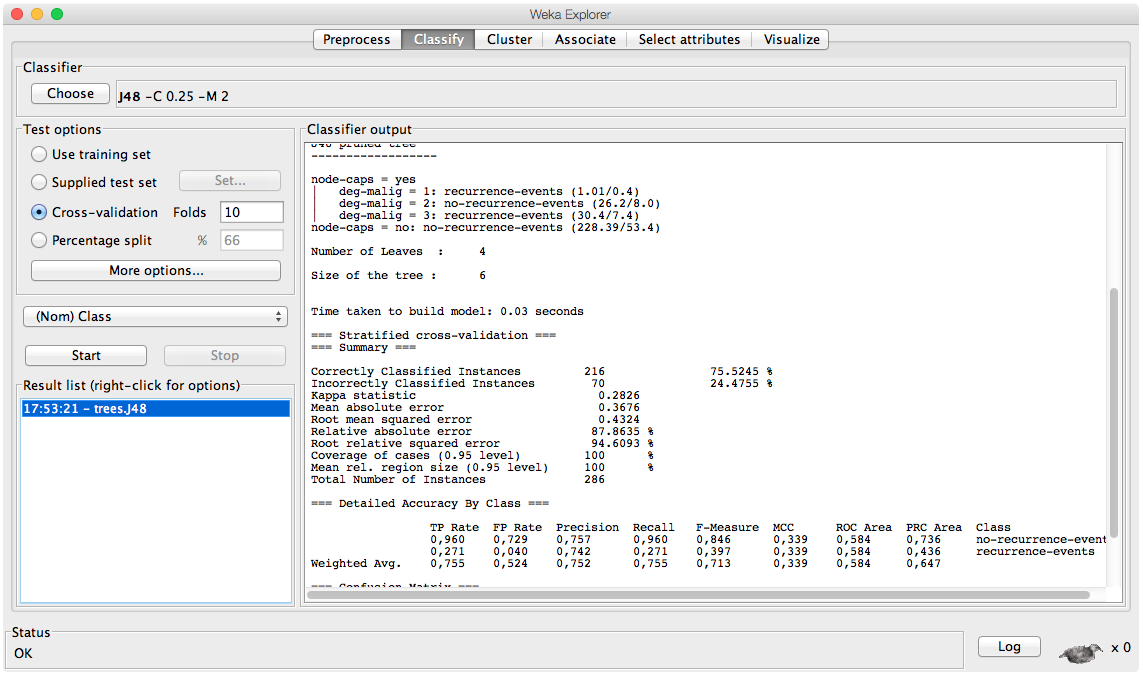

Figure 3.8 – Capture d'écran du logiciel Weka

Weka offre de nombreuses méthodes et algorithmes de data mining ainsi qu'une interface pour préparer les données à l'exploration (phase de *préprocessing*). L'outil permet d'exécuter pratiquement l'ensemble des modèles d'algorithmes cités dans ce document mis à part ce qui concerne la recherche de séquences (*Sequence Discovery*). Weka propose par défaut une série d'algorithmes classés par modèles (classification, clustering, etc.) mais de nombreux autres algorithmes sont disponibles au téléchargement via le menu de l'outil. En plus de fournir des indications textuelles, il est capable de fournir une représentation graphique des données sous forme d'arbres ou d'histogrammes en fonction de ce qui a été choisi.

Quelques points forts ressortent surtout par la façon dont l'outil a été élaboré.

Ainsi, Weka :

— est disponible gratuitement sur le site officiel ;
— offre une portabilité presque totale en raison de son implémentation Java ;
— contient un large panel d'algorithmes ainsi qu'une interface pour le prépro-cessing ;
— propose des options pour visualiser les données ;
— est facile à utiliser de par son interface graphique simple ;
— peut être utilisé en tant que librairie dans un projet.

R Le deuxième logiciel sur lequel il est intéressant de s'attarder est R [60]. Utilisé plus à des fins de statistiques (modélisation linéaire et non linéaire, tests statistiques classiques, analyses de séries temporelles, etc.) et à la création de graphiques, R permet d'utiliser un certain nombre d'algorithmes liés au data mining. Cela est possible par le caractère hautement extensible du logiciel qui offre la possibilité de télécharger près de 6.000 packages qui augmentent grandement le nombre de fonctionnalités dans R [61].

Les packages disponibles couvrent tous les aspects du data mining allant de la classification au clustering en passant par l'analyse de textes (*Sequence Discovery*). La force du logiciel a été reconnue dans KDnuggets [5] comme meilleur logiciel d'analyse, de data mining et de data science [6] pour les années 2011 à 2015. [61].

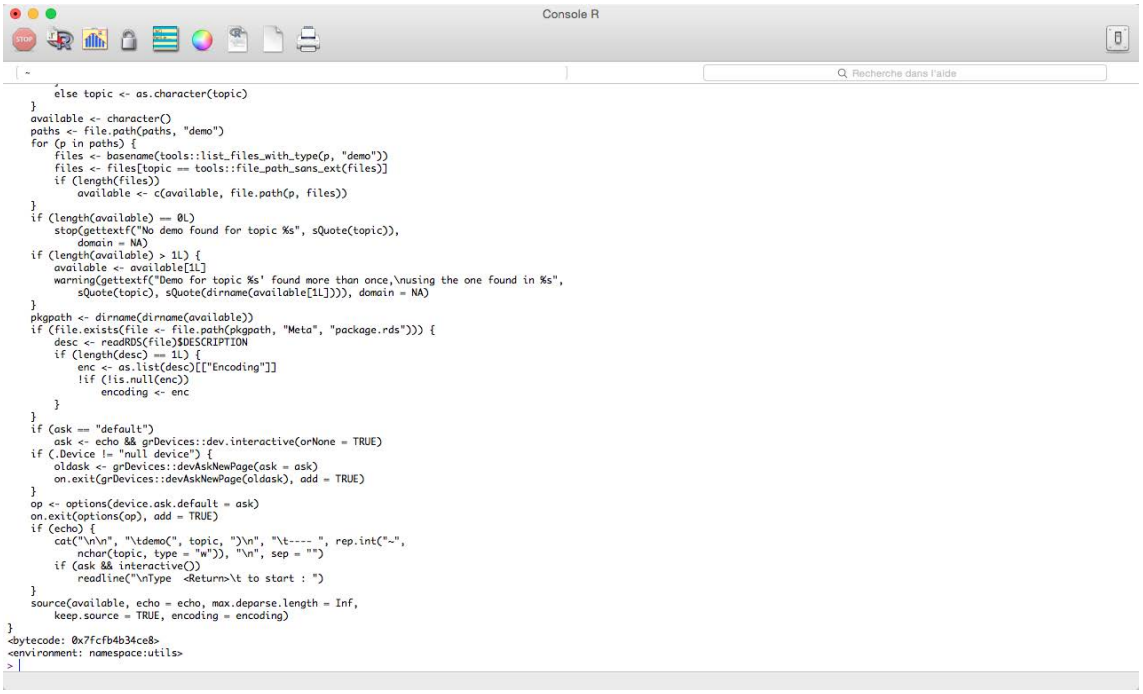

Figure 3.9 – Capture d'écran du logiciel R

5. KDnuggets réalise, entre autres, des comparaisons de logiciels sur le data mining.
6. La science des données est une nouvelle discipline qui, comme le data mining, vise à extraire la connaissance de données. Cette discipline est influencée par de nombreuses disciplines comme les mathématiques ou les statistiques.

Le logiciel, dont une capture d'écran est proposée à la figure 3.9, est un outil puissant en ce qui concerne le data mining. R est composé d'un certain nombre d'avantages mais aussi d'inconvénients. L'outil :

— est disponible gratuitement ;
— offre la possibilité de télécharger du contenu supplémentaire (6.000 packages) ;
— fonctionne sur de nombreux systèmes d'exploitation ;
— est utilisable par code R (lignes de commandes) sans interface graphique ;
— est inutilisable dans un environnement de programmation comme Eclipse[7].

Outre la présentation de ces deux logiciels très connus de l'industrie et du monde académique [61], d'autres logiciels existent mais n'ont pas été développé dans ce document. Weka et R restent les deux outils les plus intéressants au niveau du data mining.

3.4 Data Mining dans le Domaine Hospitalier

Le data mining dans le milieu hospitalier est une discipline qui a pris énormément d'ampleur depuis la dernière décennie. D'ailleurs, la discipline est majoritairement employée dans le domaine des soins de santé [62]. Ce domaine, plus que les autres, est d'autant plus critique qu'il touche la vie de personnes. Rechercher de l'information parmi des gigantesques amas de données devient vital. Il est donc nécessaire d'employer des méthodes et des algorithmes pour examiner ces données.

Les nouveaux systèmes de gestion hospitalière qui intégreraient des algorithmes de data mining pourraient prévenir les erreurs médicales mais aussi détecter au plus tôt les risques de maladies [62]. Le diagnostic assisté par un système informatique pourrait à terme devenir une réalité. Les logiciels qui intégreraient le data mining seraient capables, par exemple, de :

— Identifier les facteurs de risque de cancer de la prostate à partir des variables cliniques et démographiques [55] ;
— Prédire si un patient, hospitalisé en raison d'un infarctus, aura une deuxième crise cardiaque [55] ;
— Détecter les risques de cancer à partir d'images [63] ;
— Examiner la corrélation entre plusieurs études cliniques [64] ;
— Etc.

Bien qu'étant une discipline assez jeune dans le domaine médical, l'utilisation du data mining devient une évidence. Coupler un système informatique à l'exploration de données serait tant utile au médecin pour ses décisions qu'au patient.

7. En tout cas, aucune librairie de R n'est disponible. Il y a moyen d'intégrer une console du logiciel dans Eclipse mais il est impossible de l'utiliser dans le code Java d'un projet

Deuxième partie

Implémentation

Chapitre 4

Définition de l'Infrastructure

Le présent chapitre traite la façon dont le logiciel RetroStudy a été conçu à travers son architecture logicielle mais aussi son intégration avec les bases de données et les ontologies ainsi que les interactions avec les utilisateurs. Afin de se remémorer le contexte, la première section tentera d'expliquer la finalité du logiciel pour ensuite énoncer l'architecture. L'architecture globale, présentée à la section 4.2, décrira en détails les liens qui existent entre les différents composants cités précédemment. Pour clôturer la section, un schéma général fera la synthèse sur l'architecture choisie. Enfin, la section 4.3 présentera les technologies utilisées durant la phase d'implémentation.

4.1 RetroStudy

Cette section présente en quelques mots le logiciel qui a été conçu, avec ses avantages et une description assez abrégée des fonctionnalités. Elle vise à poser le contexte de l'application et dégrossir la façon dont RetroStudy a été implémenté. Il s'agit d'un prélude aux chapitres suivants.

4.1.1 Présentation Générale

RetroStudy vise à aider les médecins dans leurs décisions vis-à-vis des études cliniques. Il n'est pas toujours facile de connaître les relations entre certaines maladies ou tout simplement de faire des liens entre celles-ci. Le logiciel a l'ambition de fournir un système de visualisation de données basé sur des contraintes élaborées à la demande.

Le genre d'application qui a été choisi est un site web. Contrairement à une application standalone, une application sur le web donne lieu à une série d'avantages. Ceux-ci sont décrits ci-dessous :

- **Accessibilité optimale.** Tout d'abord, une application web permet une portabilité quasi totale. En effet, un navigateur suffit pour accéder à l'information. De ce fait, un utilisateur qui utilise son ordinateur professionnel, personnel ou même son smartphone, peut accéder à l'information.

- **Maintenabilité simplifiée.** Une application unique a l'avantage pour la maintenance. Lorsque de nouvelles fonctionnalités sont créées ou que l'interface graphique est modifiée, les changements sont disponibles pour l'ensemble des utilisateurs au même moment.

- **Traitement partagé.** La charge de calcul est dispatchée entre la machine de l'utilisateur et le serveur. Généralement, le navigateur de l'utilisateur se contente de réaliser les affichages alors que le serveur réalise les calculs plus lourds.

- **Système d'historique.** Les opérations propres aux utilisateurs sont enregistrées et centralisées sur le serveur. Ainsi, peu importe la machine que l'utilisateur utilise, celui-ci aura toujours accès à ses données ou ses fichiers précédents.

- **Rapidité de création.** Les feuilles de style CSS offrent de nombreuses possibilités pour créer des interfaces graphiques assez complexes. La création de pages HTML en est grandement simplifiée.

4.1.2 Fonctionnalités

RetroStudy est composé d'un ensemble d'éléments que les utilisateurs peuvent employer. Les fonctionnalités qui ont été implémentées sont exposées tout au long de cette sous-section.

Bien que le choix des fonctionnalités doive faire l'objet d'une démarche d'ingénierie des exigences, il n'a pas été évident de faire intervenir les médecins, principaux intéressés, dans cette démarche. Leur agenda fort chargé n'a donc pas permis la réalisation d'une étude complète des fonctionnalités à développer.

Accès sécurisé L'application web offre une interface de connexion. Seul un utilisateur enregistré peut avoir accès aux fonctionnalités. Les données manipulées par le logiciel sont considérées comme confidentielles. Dès lors, n'importe qui ne peut voir l'information. Pour une meilleure sécurité, les données échangées sur le web sont chiffrées grâce au protocole HTTPS.

Statistiques sur l'utilisateur Une fois l'utilisateur connecté, celui-ci peut retrouver des indicateurs chiffrés sur ses précédentes requêtes. Premièrement, il retrouve le nombre d'études cliniques en cours dont il est le responsable (i.e. investigateur principal). Ensuite, l'utilisateur peut découvrir le nombre de requêtes soumises via le formulaire en comparaison avec ses collègues du même hôpital. Enfin, la possibilité d'émettre le logiciel au niveau de tous les hôpitaux belges ouvre la porte à une autre statistique : le nombre d'études cliniques (d'un établissement) par rapport aux autres hôpitaux belges.

Envoi d'une requête Fonctionnalité principale de RetroStudy, la création et l'envoi d'une requête se déroule via un formulaire dynamique. Le côté dynamique donne la possibilité à l'utilisateur de créer une requête à la demande. Les champs du formulaire autorisent l'utilisateur à poser des contraintes sur certaines caractéristiques des patients. Par exemple, il arrive qu'un médecin veuille connaître les résultats d'une requête simple comme celle-ci :

Quels sont les patients de sexe masculin de moins de 40 ans et de plus de 18 ans ayant une glycémie élevée ?

Le formulaire s'adapte en fonction du choix de l'utilisateur. Il s'agit donc de réaliser une contrainte sur le sexe du patient, une autre sur son âge et enfin une dernière sur ses données biologiques. L'application transforme les contraintes en une requête SQL unique plus ou moins complexe.

Data mining Avant de récupérer les résultats de sa requête, l'utilisateur peut choisir la méthode de data mining qui lui convient. Pour l'heure, seule la méthode d'arbres de décisions (*Random Trees*) est fonctionnelle. L'idée serait d'étendre cette fonctionnalité à d'autres méthodes comme le *clustering* par exemple. Le data mining, au niveau du présent logiciel, permet de faire ressortir certaines statistiques mais aussi une représentation schématique sous forme d'arbre (dans le cas des arbres de décisions), générée par la librairie Weka (voir section 4.3 et chapitre 6).

Liste des patients Pour pouvoir pleinement réaliser la tâche de prise de contact avec les patients éligibles à une étude clinique, le logiciel présente une liste de patients qui correspondent à la requête émise. Les informations nécessaires de contact s'y retrouvent comme l'adresse et leur numéro de téléphone. Cette liste de patients peut également être imprimée pour plus de confort.

Visualisation des données En fonction de la requête émise, une fonctionnalité de visualisation dynamique des données a été mise en place. Les contraintes sont regroupées en catégories : informations patients, biologie clinique, etc. Suivant que l'utilisateur ait posé une contrainte dans une des catégories, l'ensemble des valeurs pour toutes données contenues dans cette catégorie est proposé sur le graphique. Par exemple, la requête mise en évidence ci-dessus permettrait de récupérer les données sur la glycémie mais également le nombre de globules blancs, de globules rouges, le taux d'urée, etc. Le graphique peut alors être construit à la demande. Il suffit de choisir les données à affecter aux différents axes pour pouvoir les comparer.

Historique des requêtes Toutes les requêtes réalisées par un médecin via l'application web sont enregistrées sur le serveur. Ainsi, l'utilisateur a accès à un historique complet de ses requêtes, des patients correspondants, des statistiques générées et au système de visualisation. Une option permet également de reconduire la requête dans le cas où de nouveaux patients seraient éligibles.

4.2 Architecture Détaillée

Dans cette section, l'architecture complète du logiciel et les différents composants en lien avec RetroStudy sont détaillés. Les sous-sections suivantes présentent chaque composant interagissant avec l'application créée (*Serveur Applicatif*). Un schéma intermédiaire est construit au fur et à mesure de la description. Le schéma complet clôturera la section.

4.2.1 Base de Données

Pour accéder aux valeurs des examens biologiques réalisés sur des patients, le logiciel à besoin d'une connexion avec une base de données. Les données nécessaires sont stockées suivant le standard vMR (voir chapitres 1 et 5). Une fois la connexion ouverte entre l'application et la base de données, les éléments peuvent être échangés. Le processus classique de récupération d'informations tient en deux étapes : l'envoi d'une requête SQL et la récupération d'un résultat.

La base de données et le programme RetroStudy ont l'avantage de fonctionner sur le même serveur physique. Cela a pour conséquence d'améliorer considérablement la rapidité des échanges de données entre les deux entités. Le schéma A.4 illustre ces propos.

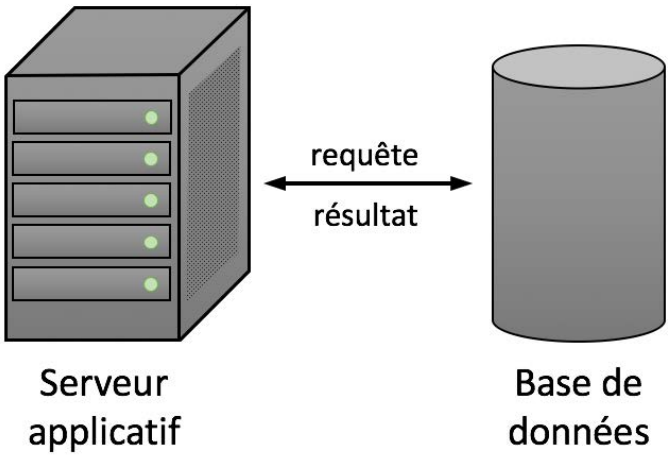

Figure 4.1 – Illustration du lien entre l'application et les données

4.2.2 Ontologies

Comme il en a déjà été discuté dans la première partie du document, la puissance des bases de données relationnelles est assez limitée. Pour améliorer le nombre de résultats d'une requête classique, un système d'ontologies devait être mis en place. Il n'a pas encore été possible de connecter les ontologies avec le programme RetroStudy. Malgré ce contretemps, le système global a été étudié pour pouvoir coupler facilement les ontologies à la base de données. Ce couplage donne lieu à une nouvelle catégorie de base de données : les bases de données sur le web sémantique.

La plateforme D2RQ est un système qui propose de considérer une base de données relationnelle comme un graphe « virtuel » RDF, à travers lequel on peut accéder au contenu de la base, en utilisant des API telles que Jena [1] [45]. Afin d'utiliser la base de données basée sur le web sémantique, il faut coupler les ontologies à la base de données. Pour réaliser cette intervention, un ou plusieurs fichiers de mapping sont nécessaires. Ces types de fichiers sont précisément générés par la plateforme D2RQ.

Une représentation schématique est disponible à la figure 4.2.

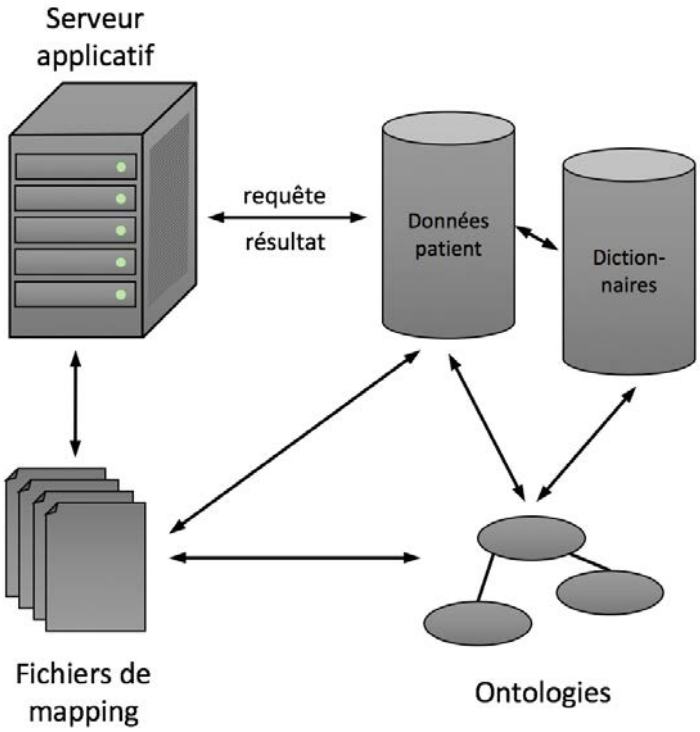

Figure 4.2 – Illustration du lien entre les ontologies, les données et l'application

1. Jena est une librairie Java qui permet de créer des ontologies, des graphes RDF et poser des requêtes SPARQL.

4.2.3 Serveur d'Applications

Comme il en a déjà été question, le logiciel développé est une application web. Cette dernière a ceci de particulier : elle comporte deux types de langages : le Java (généralement) et le HTML. A travers cette application, une bonne architecture est nécessaire afin de minimiser les coûts de maintenance. Quelles sont les caractéristiques de l'architecture logicielle ?

MVC Le projet web fait appel à un modèle de conception (*design pattern*) qui définit une bonne pratique à adopter dans la création de logiciels. Le premier pattern utilisé est le *Modèle-Vue-Contrôleur* (MVC). Il découpe l'application en couches distinctes et a un impact significatif sur l'organisation du code.

Théoriquement, le pattern MVC se découpe en trois parties :

— tout ce qui concerne le traitement, le stockage et la mise à jour des données de l'application doit être contenu dans la couche « Modèle » ;
— ce qui concerne l'interaction avec l'utilisateur et la présentation des données (mise en forme de l'affichage) doit être contenu dans la couche « Vue » ;
— ce qui se rapporte au contrôle des actions de l'utilisateur et des données doit être contenu dans la couche nommée « Contrôle ».

Dans le *modèle*, nous trouvons les données et les traitements à appliquer aux données. Dans cette partie, se retrouvent les objets Java d'une part, qui peuvent contenir des attributs (données) et des méthodes (traitements) qui leur sont propres, et un système capable de stocker des données d'autre part.

Dans la *vue*, il y a des pages JSP [2]. Les pages s'exécutent côté serveur et permettent l'écriture de gabarits (pages en langage « client »).

Dans le *contrôleur*, ce sont des servlets qui y sont contenus. Les servlets sont des objets qui permettent d'intercepter les requêtes faites par un client, et qui peuvent personnaliser une réponse en conséquence. Ils fournissent pour cela des méthodes permettant de scruter des requêtes HTTP. Ces servlets n'agissent jamais directement sur les données, il faut les voir comme de simples aiguilleurs : ils interceptent une requête issue d'un client, appellent éventuellement des traitements effectués par le *modèle*, et ordonnent en retour à la *vue* d'afficher le résultat au client.

DAO Le pattern DAO (*Data Access Objects*) permet de faire le lien entre la couche d'accès aux données et la couche métier d'une application (les classes Java). Le modèle DAO propose de regrouper les accès aux données persistantes dans des classes séparées, plutôt que de les disperser dans le code. Il s'agit surtout de ne pas écrire ces accès dans les classes métiers, qui ne seront modifiées que si les règles de gestion changent.

2. Les pages JSP sont des pages HTML dans lesquelles du code Java peut être inséré. Ces pages sont peu différentes des pages HTML et n'apportent pas de grande complexité. Dans ce projet web, les tags propres aux JSP n'ont pas été utilisés. Supposons donc qu'il s'agit de pages HTML

Ce modèle complète le pattern Modèle-Vue-Contrôleur utilisé qui préconise de séparer dans des classes différentes les problématiques : les vues (charte graphique, ergonomie), le modèle (cœur métier) et les contrôleurs (tout le reste : l'enchaînement des vues, les autorisations d'accès, etc.).

La mise en place du pattern est très simple. Pour chaque table de la base de données, une interface reprend toutes les méthodes pour avoir accès aux données. Ensuite, une classe « implémentation » implémente les méthodes définies dans l'interface.

L'utilisation de DAO permet de s'abstraire de la façon dont les données sont stockées au niveau des objets métiers. De cette manière, le changement du mode de stockage ne remet pas en cause le reste de l'application.

Business Objects Les objets Java permettent de manipuler les éléments récupérés de la base de données. Ces objets sont différents des objets du pattern DAO, qui ne manipule que les objets présents dans la base de données. Par exemple, il est plus aisé de manipuler les données du formulaire des contraintes de cette façon. Des méthodes Java permettent le passage du formulaire à l'objet.

Partie algorithmique Dans la partie algorithmique se trouvent les différents algorithmes utilisés pour les calculs ou les manipulations de fichiers. Différentes classes sont utilisées pour des opérations sur les dates, pour la création et la manipulation de fichiers XML. Ces fichiers représentent les résultats des requêtes faites par l'utilisateur mais également les opérations de data mining.

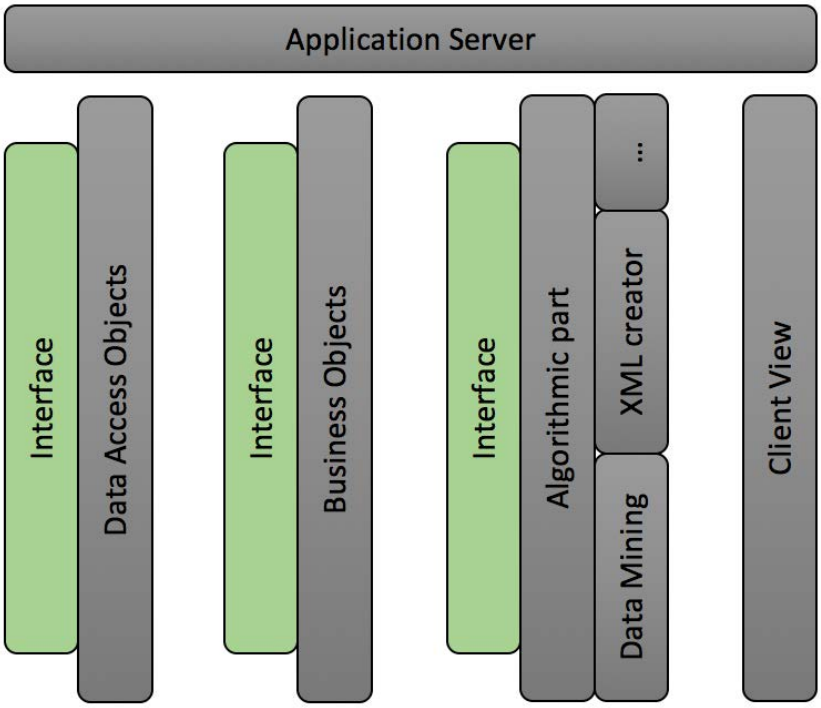

Figure 4.3 – Illustration de l'architecture du logiciel

Schéma de l'architecture logicielle La figure 4.3 représente l'architecture de la partie logicielle de l'application. Les éléments cités précédemment s'y retrouvent au niveau du serveur applicatif. De la gauche vers la droite se trouvent : le pattern DAO, les objets business, la partie algorithmique et enfin le pattern MVC illustré par la vue client. Bien entendu, le code a été agencé suivant des packages qui regroupent ces patterns.

4.2.4 Vue Globale

En guise de récapitulatif, le schéma global (figure 4.4) illustrant l'architecture physique du projet dans sa totalité est représenté. Il réalise la synthèse de tous les éléments décrits dans les paragraphes précédents. Basiquement, l'utilisateur se connecte à l'application via l'adresse du site web. Toutes les requêtes faites par l'utilisateur sur l'application transitent par le serveur applicatif qui contient une instance de l'application web et de la base de données patients. Il était aussi prévu de coupler des dictionnaires médicaux à la base de données patients. Les ontologies, quant à elles, sont liées aux données par des fichiers de mapping.

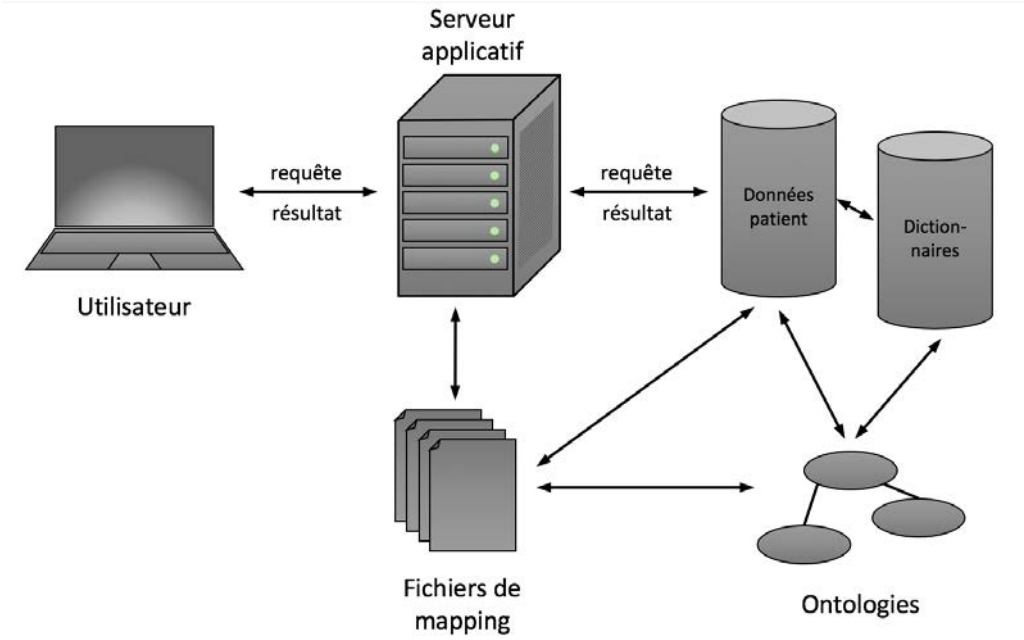

Figure 4.4 – Illustration de l'architecture globale visée

4.3 Technologies

La section *Technologies* présente les technologies utilisées durant la phase d'implémentation de l'application web. Il ne s'agit pas de décrire le fonctionnement de chaque élément en détail mais plutôt de survoler les technologies ou les librairies utilisées dans le projet. Certaines d'entre elles feront d'ailleurs l'objet d'une section dédiée plus loin dans ce document.

Java Le langage utilisé pour coder la partie business de l'application est le Java [65]. La version du langage est l'édition SE 1.8. Le Java a ceci de particulier : il s'agit de la plateforme la plus répandue dans le monde [65]. Elle a été conçue pour permettre aux développeurs de coder des applications qui peuvent fonctionner sur quasi tous les environnements possibles. De ce fait, Java donne la possibilité d'une interopérabilité améliorée avec d'autres logiciels. Dans le monde, de très nombreux modules ou librairies sont également développés en Java. La facilité de développement, l'interopérabilité et la facilité d'évolution ont été les principaux critères de sélection.

JavaScript Dans la partie affichage de la page web à l'écran, le HTML est le principal langage utilisé. Pour augmenter l'expérience utilisateur mais aussi pour définir des méthodes de calculs, un autre langage est nécessaire : le Javascript. Ce langage est tout à fait différent de Java. Le code en Javascript est directement inclus dans les pages HTML. Une action possible est, par exemple, d'ajouter du dynamisme à certains composants de la page, comme des graphiques.

Bootstrap Afin de ne pas développer un site web à partir de rien, des composants existent pour faciliter la vie des développeurs. Les feuilles de styles CSS contiennent du code à réutiliser pour agencer les éléments sur les pages web. Le framework complet Bootstrap les intègre [66]. Il s'agit en réalité du thème utilisé par le site `http://www.twitter.com`.

Highcharts La librairie Highcharts facilite l'intégration de graphiques dans des documents web [67]. Elle dispose de nombreuses possibilités d'intégration de graphes allant des graphes en bâtons jusqu'au cartes des pays du monde. Les données nécessaires à la création de tels graphiques doivent être envoyées en JSON. Ce langage, très simple, n'est pas discuté dans ce document. Une méthode de transformation des données brutes a simplement été développée.

Weka Les méthodes d'exploration de données ont été fournies par la librairie Weka [59]. Celle-ci a en son sein des fonctions prédéfinies comme les arbres aléatoires, le clustering, etc. Pour plus d'informations, lire les chapitres 3 et 6 dédiés au data mining.

GraphViz Une petite librairie a également été utilisée pour la création de graphes. Il s'agit de la librairie GraphViz [68]. Elle a permis de créer les représentations des arbres illustrant les données pertinentes qui ont été préalablement classées par les méthodes de data mining. La visualisation des données rend la vision des résultats plus confortable.

Chapitre 5

Création d'une Base de Données

Ce chapitre présente comment la base de données a été construite et adaptée à partir d'un modèle préconçu mais aussi les différentes étapes nécessaires à son achèvement. Ce qui est écrit ici est à mettre en relation avec le chapitre *Database* présenté dans la première partie de l'ouvrage au chapitre 1. Les compromis réalisés ainsi que les phases de la construction seront mis en avant à travers les différentes sections de ce chapitre.

5.1 Choix du Modèle

L'entreprise MIMS, qui est partenaire dans le projet IFLEC, utilisait déjà massivement les bases de données relationnelles dans les produits qu'elle mettait à disposition de ses clients du milieu hospitalier. La volonté de créer une application efficace passait par l'utilisation d'une base de données. Plusieurs choix étaient possibles comme garder une base de données relationnelle, stocker les données sous forme de fichiers XML ou encore envisager d'utiliser un système comme NoSQL.

Il a été choisi de continuer dans la voie des bases de données relationnelles car MIMS utilisait déjà le serveur de bases de données Oracle. Adopter une base de données relationnelle Oracle était conseillé surtout si les données devaient être migrées vers RetroStudy mais aussi par facilité dans la séparation des tâches dans IFLEC. La préférence pour un modèle universel et standardisé se faisait sentir car c'était un avantage, surtout pour uniformiser les données stockées entre les différents hôpitaux. Le choix s'est donc porté tout naturellement vers le modèle vMR (*Virtual Medical Record*). Ce modèle reste néanmoins en phase de développement et des modifications restent possibles pour l'organisation des concepts dans les schémas de vMR.

5.2 Schémas vMR Initiaux et Code SQL Généré

Avant de commencer la transformation proprement dite, cette section vise à donner une base pour la compréhension de la transformation. La première sous-section réalise un bref rappel du modèle vMR du point de vue de sa représentation tandis que la deuxième montre l'utilisation d'un logiciel spécifique utilisé pour la génération automatique de code.

5.2.1 Vue Générale des Schémas vMR

HL7 a publié une documentation détaillée de son modèle vMR [14]. Dans cette documentation, non seulement une explication générale du modèle est fournie mais aussi une partie largement consacrée aux schémas. La documentation et le modèle en lui-même décomposent la difficulté en créant une multitude de schémas qui présentent chacun un concept du domaine médical. Au lieu d'avoir un schéma général, celui-ci est découpé en concepts décrivant un domaine médical particulier. Par exemple, la figure 5.1 décrit le concept *Substance*.

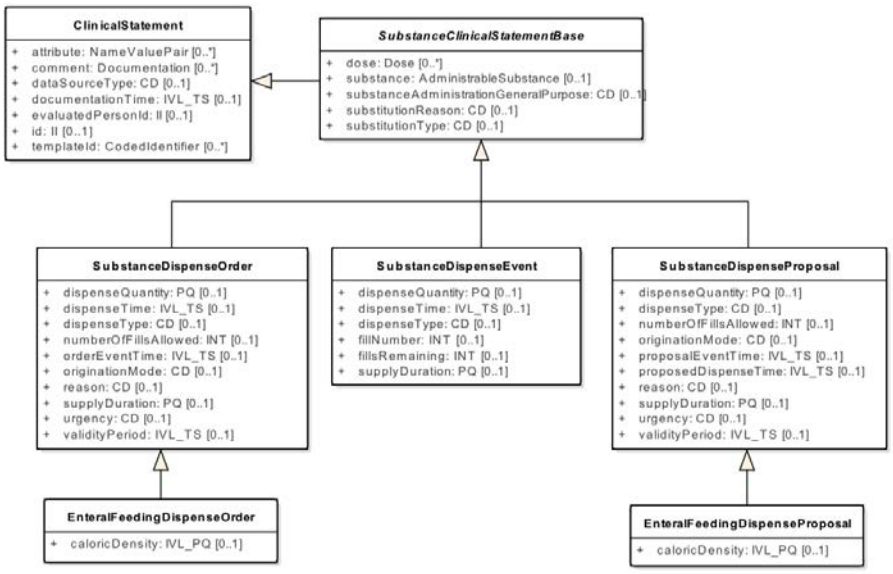

Figure 5.1 – Concept de *Substance* inclus dans le modèle vMR

5.2.2 Génération du Code SQL

Le modèle vMR a été téléchargé à partir du site web de HL7 en tant qu'archive. Elle contenait le document d'information mais aussi un fichier .eap. Ce genre de fichier peut être ouvert à l'aide du programme Enterprise Architect développé par SPARX Systems [69]. L'utilisation du programme est réservée aux professionnels qui désirent utiliser toute la puissance de la modélisation pour la création d'une application. Durant la phase de transformation, le logiciel a été utilisé dans sa version d'évaluation.

Dans le cas présent, vMR, une fois chargé dans Enterprise Architect, peut être manipulé à travers une multitude de fonctionnalités dont celle qui permet de visualiser les données sous la forme de schémas, à l'instar de ceux présents dans la documentation. Les fonctionnalités qui sont utiles dans ce travail de création de base de données est l'export à proprement parler. Le logiciel autorise non seulement les modifications des types d'entités et des relations entre celles-ci mais aussi une transformation directe des schémas vers du code SQL propriétaire comme Oracle ou MySQL.

Vu la construction du modèle, certains éléments, comme par exemple l'utilisation quasi systématique des relations *is-a*, devront être modifiés. Deux possibilités s'offraient à nous pour la transformation : effectuer directement les modifications nécessaires dans les schémas vMR à l'aide du logiciel Enterprise Architect ou générer le code SQL et ensuite effectuer les transformations d'une autre manière. La deuxième proposition a été privilégiée car les manipulations sur Enterprise Architect sont complexes, lentes et approximatives.

Code 5.1 – Exemple de code SQL généré par Enterprise Architect

```
1  CREATE TYPE SubstanceDispenseOrder UNDER SubstanceClinicalStatementBase
2  (
3      dispenseQuantity            PQ,
4      dispenseTime                IVL_TS,
5      dispenseType                CD,
6      numberOfFillsAllowed        INT,
7      orderEventTime              IVL_TS,
8      originationMode             CD,
9      reason                      CD,
10     supplyDuration              PQ,
11     urgency                     CD,
12     validityPeriod              IVL_TS,
13     substanceDispenseOrderID    Integer NOT NULL
14 );
```

Le code 5.1 montre un exemple de code SQL généré par le logiciel. Bien que ce code soit valide, quelques soucis peuvent apparaître si ce code SQL est utilisé tel quel. Plusieurs transformations seront donc nécessaires. Les détails concernant ces problèmes engendrés par le modèle vMR ainsi que les transformations qui les concernent sont présentés dans la section suivante.

5.3 Transformation du SQL Généré

Cette sous-section traite de la problématique de la transformation du SQL généré par le logiciel Enterprise Architect vers le SQL final. Ce dernier sera utilisé pour créer notre instance de base de données. Dans la suite, les problèmes potentiels seront présentés en premier suivis des corrections apportées au code généré.

5.3.1 Problèmes Potentiels

Quelques problèmes peuvent survenir au vu du schéma vMR. Une analyse des schémas et du code SQL généré a permis de mettre au jour certains cas problématiques dont les principaux sont présentés ci-dessous :

Types de données Le modèle vMR est construit d'une manière hiérarchique où les relations *is-a* sont particulièrement récurrentes. D'ailleurs, les relations (classiques) entre deux entités ne sont présentes que huit fois à travers les différents schémas du modèle. Les types de données au niveau des bases de données permettent une certaine abstraction et simplification. Pour reprendre l'exemple de *Substance* déjà présenté (code 5.1), l'attribut *dispenseQuantity* est de type *PQ*. Cet attribut est dit complexe car il ne fait pas référence à un type primitif que sont les chaînes de caractères ou les nombres. Cette façon d'écrire simplifie grandement le code et permet la réutilisation des types par d'autres attributs. Malgré cet avantage indéniable, le code SQL du modèle vMR ne compte que des types de données et aucune table. Or, pour pouvoir stocker des données efficacement dans une base de données, il est nécessaire d'avoir des tables. Une illustration est présentée au code 5.2 pour faire la synthèse de ces propos.

Code 5.2 – Présentation du type de données d'un attribut (en SQL)

```
1  -- un attribut de SubstanceDispenseOrder et son type
2
3      dispenseQuantity            PQ
4
5
6  -- le type PQ lié à l'attribut dispenseQuantity
7
8  CREATE TYPE PQ AS OBJECT
9  (
10     unit    Code,
11     value   Decimals,
12     pQID    Integer
13 );
```

Hiérarchie des types de données Alors que les types de données posent quelques problèmes, les relations entre les entités (i.e. les types de données), schématisées par les relations *is-a*, sont aussi une source de difficultés. L'embarras de ce genre de relation se mesure en deux points. Premièrement, tous les types de données de vMR sont sous-types ou sur-types les uns des autres. Il en ressort une interconnexion assez complexe. Deuxièmement, certains serveurs de base de données (SGBD) modifient d'eux-mêmes les types en tables pour pouvoir les utiliser. Certains types, dus à l'héritage, sont transformés en une table de plus de 1000 colonnes. Cela est beaucoup trop lourd pour le SGBD qui ne peut réaliser le traitement.

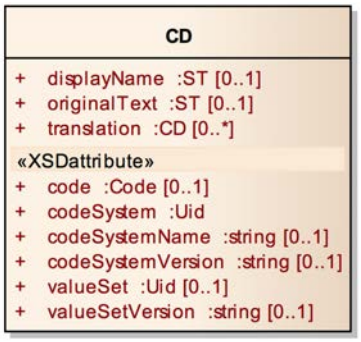

Figure 5.2 – Illustration de *CD* et de la récursivité sur *translation*

Récursivité des types de données Une autre critique qui peut être émise est la récursivité des types. Certains types du schéma d'origine (figure 5.2) font apparaître une récursion dans leur relation. Une partie du code généré par Enterprise Architect permet de montrer, via le code 5.3, le problème rencontré. L'attribut concerné serait à chaque fois un type *CD* qui contiendrait le même attribut de type *CD*. La récursion s'arrêterait si, à un moment donné, *translation* était *null*. Le schéma hiérarchique, lui, indique que *translation* est de cardinalité [0-n]. Alors que le code indique une cardinalité [0-1], il est clair que le code et le schéma mis en parallèle ne concordent pas.

Code 5.3 – Exemple de la récursion dans un type (en SQL)

```
1  -- Un attribut de CD est de type CD
2
3  CREATE TYPE CD AS OBJECT
4  (
5      code                 Code,
6      codeSystem           id,
7      codeSystemName       string,
8      codeSystemVersion    string,
9      displayName          ST,
10     originalText         ST,
11     translation          CD,
12     valueSet             id,
13     valueSetVersion      string,
14     cdID                 Integer
15 );
```

Les énumérations Le modèle vMR a défini des types énumérés où chaque type énuméré est à considérer comme un ensemble de valeurs exhaustives. La nécessité de supprimer tous les types du code SQL oblige une adaptation des types énumérés. Ceux-ci ne peuvent d'ailleurs pas être utilisés par un SGBD tels qu'ils ont été générés au départ. Le type présenté à la figure 5.3 montre l'énumération de valeurs pour le concept de télécommunication.

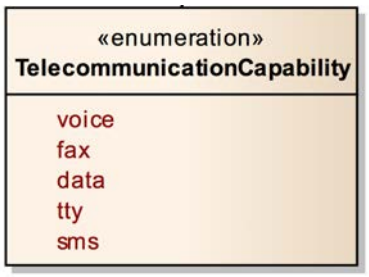

Figure 5.3 – Type énuméré du concept de télécommunication

5.3.2 Correction des Problèmes

Afin de pallier aux problèmes relevés dans la sous-section précédente, une méthodologie de correction a été établie. Elle permet de transformer, étape par étape, le code SQL généré en code SQL final pour en supprimer les difficultés potentielles. La méthodologie est décrite suivant un ordre chronologique et n'a pour le moment pas été automatisée.

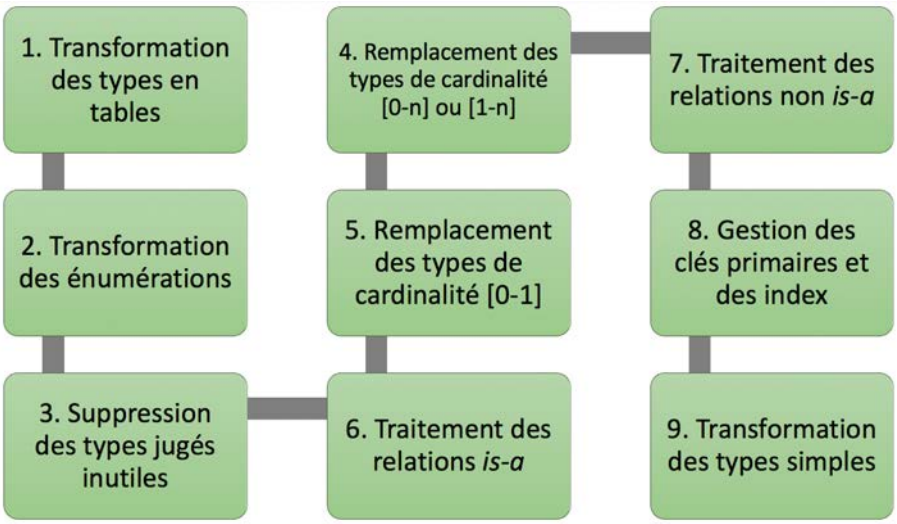

Figure 5.4 – Méthodologie de transformation vers le SQL final

Comme la figure 5.4 le montre, la méthodologie est composée de 9 étapes. Ces différentes phases ont été réalisées dans cet ordre lors du projet RetroStudy. Voici les détails concernant les étapes de la transformation du SQL généré par Enterprise Architect vers le SQL final qui sera importé par le SGBD.

1. Transformation des types en tables

La première étape n'est pas compliquée en soi. Deux cas sont à prendre en compte : le cas où un type hérite d'un autre et le cas où le type est supertype c'est-à-dire qu'il n'a aucun parent. La manière de transformer varie d'un cas à l'autre mais le résultat final est identique. L'idée est de supprimer les mots-clés *UNDER* et *AS OBJECT* et de remplacer le mot-clé *TYPE* par *TABLE*. Les transformations liées aux hiérarchies des types (relations *is-a*) sont détaillées à la sixième étape de la méthodologie.

Code 5.4 – Illustration de la première étape de transformation

```
1 -- Cas 1 : héritage d'un autre type
2 CREATE TYPE SubstanceDispenseOrder UNDER SubstanceClinicalStatementBase
3
4 -- Cas 2 : supertype
5 CREATE TYPE CD AS OBJECT
6
7 -- Résultat obtenu après transformation
8 CREATE TABLE SubstanceDispenseOrder et CREATE TABLE CD
```

2. Transformation des énumérations

Les changements à effectuer permettent d'une part d'éviter l'utilisation des types de données et d'autre part de garder les contraintes sur les valeurs à utiliser. Comme l'énumération n'était pas possible via le SGBD, il a fallu trouver le moyen de garder la référence vers les valeurs possibles. Après la première étape de transformation, le code n'est pas suffisant. Des modifications s'imposent.

Les valeurs d'énumération sont transformées en un identifiant de la table dont le type est une chaîne de caractères (*varchar2*). Pour compléter l'identifiant, un label et une description ont été ajoutés. Le label définit le texte qui correspond à la valeur d'énumération. La description résume en quelques mots la signification de la valeur courante. Elle fait partie de la documentation de vMR. Ainsi, pour *voice* par exemple, l'identifiant de table serait *voice*, le label *Voice* et la description *This device can receive voice calls*. La transformation passe donc par un stade de création d'une table à 3 attributs et un stade d'insertion de valeurs. Le code 5.5 donne une indication sur la manière de réaliser cette deuxième phase de transformation.

Code 5.5 – Illustration de la deuxième étape de transformation

```
1  -- Code SQL généré par Enterprise Architect
2
3   CREATE TYPE TELCAPABILITY AS OBJECT
4  (
5    value ENUM('voice','fax','data','tty','sms')
6  ) not final;
7
8
9  -- Code SQL de transformation en table
10
11 CREATE TABLE TELCAPABILITY
12 (
13   ID              VARCHAR2(10 BYTE)    NOT NULL,
14   LABEL           VARCHAR2(20 BYTE),
15   DESCRIPTION     VARCHAR2(1000 BYTE)
16 );
17
18 -- Remplissage des données suivant la documentation vMR
19
20 Insert into TELCAPABILITY values ('voice', 'Voice', 'This␣device␣can␣receive
       ␣voice␣calls');
21 Insert into TELCAPABILITY values ('fax', 'Fax', 'This␣device␣can␣receive␣
       faxes');
22 Insert into TELCAPABILITY values ('data', 'Data', 'This␣device␣can␣receive␣
       data␣calls␣(i.e.␣modem)');
23 Insert into TELCAPABILITY values ('tty', 'Text', 'This␣device␣is␣a␣text␣
       telephone');
24 Insert into TELCAPABILITY values ('sms', 'SMS', 'This␣device␣can␣receive␣SMS
       ␣messages');
```

3. Suppression des types jugés inutiles

Lors de l'analyse détaillée du code et de la documentation, certains types définis dans le modèle vMR ne contenaient qu'un seul attribut. Cet attribut était souvent de type simple et faisait référence soit à une chaîne de caractères soit à un nombre. Garder une table d'une seule colonne ne contenant qu'une valeur nominale ou chiffrée n'avait guère de sens. Il a été décidé de simplement supprimer les tables concernées. Dès lors, il ne faut pas oublier de remplacer les occurrences du type supprimé dans le code SQL par *varchar2* ou *number*. La liste des tables supprimées est la suivante :

- HXIT
- QSET
- IVL
- BL
- CS
- ST
- INT
- REAL
- TS
- Code
- Decimal
- Uid
- TimeStamp
- Uri
- set_PostalAddressUse
- set_EntityNamePartQualifier
- set_EntityNameUse
- set_TelecommunicationAddressUse
- set_TelecommunicationCapability

Les tables supprimées, dont le préfixe est *set_*, sont en réalité liées à des fonctions qui peuvent utiliser des valeurs énumérées. Ces tables n'ont aucun attribut. C'est pour cette raison qu'il a été décidé de les supprimer. Ces tables sont tout de même en relation avec d'autres tables dans le schéma vMR. Pour faire simple, une fois que ces tables ont été supprimées, il suffit de supprimer le préfixe *set_* dans toutes les tables restantes. Cela a pour but de faire le lien avec les tables d'énumérations mais plus avec les tables représentant les fonctions. Pour plus d'informations, le lecteur est invité à lire l'annexe B.

4. Remplacement des types de cardinalités [0-n] ou [1-n]

Il n'est pas rare de devoir insérer plus d'une valeur d'un même type dans une table, que se soit des valeurs facultatives [0-n] ou obligatoires [1-n]. Pour pouvoir enregistrer plusieurs données, au contraire des attributs simples, il est nécessaire de réaliser une adaptation dans le modèle vMR et dans le code SQL. L'ingénierie des bases de données [70] a défini, au fur et à mesure de son évolution, des bonnes pratiques de transformation.

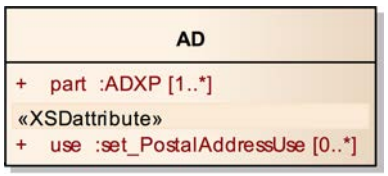

Figure 5.5 – Illustration du concept de *AD*

Pour détailler les 2 cas possibles, le concept de *AD*, faisant référence à la notion d'adresse, est exposé ici. Sur la figure 5.5, les cardinalités présentes sont [1-n] et [0-n] et concernent les situations problématiques. Le code 5.6 illustre le code SQL généré pour *AD*.

Code 5.6 – Code SQL généré et transformé pour *AD*

```sql
-- Code SQL généré par Enterprise Architect

CREATE TYPE AD AS OBJECT
(
    part    ADXP,
    use     set_PostalAddressUse,
    adID    Integer
);


-- Résultat des transformations précédentes

CREATE TABLE AD
(
    part ADXP,
    use PostalAddressUse,
    adID Integer
);
```

1. Cardinalité [0-n] : Afin de mieux comprendre la cardinalité [0-n], l'explication par l'exemple reste la meilleure manière de faire. Concernant *AD*, *use* n'a aucune limite au niveau de sa taille. De manière à stocker toutes les données possibles, une table intermédiaire doit être construite pour pouvoir contenir les informations nécessaires. Pour ce faire, la table intermédiaire doit contenir l'identifiant de la table *AD* ainsi que l'identifiant de la table qui fait référence au type de données. En l'occurence, pour cet exemple, la table intermédiaire se voit doter de *adID* (l'identifiant de la table) et de *ID* (l'identifiant de *PostalAddressUse*).

Schématiquement, une nouvelle table est créée afin de faire le lien direct entre deux tables liées par la cardinalité. La figure 5.6 illustre la création d'une table intermédiaire entre les tables *AD* et *PostalAddressUse*. De cette manière, il est possible de garantir la cardinalité [0-n]. Les données qui devaient initialement être stockées dans *AD* le sont maintenant dans *Use_AD*. Il suffit de faire correspondre les identifiants des deux autres tables. Il faut noter que l'attribut *use* de *AD* n'a plus lieu d'être. Il est tout bonnement supprimé.

Figure 5.6 – Création d'une table intermédiaire pour la cardinalité [0-n]

2. Cardinalité [1-n] : La cardinalité [1-n] reste assez proche de celle de [0-n]. Le principe reste le même : il faut créer une table intermédiaire. Pour garder l'exemple de *AD*, *part* est supprimé de cette table. Par contre, une nouvelle table est créée, faisant l'intermédiaire entre *AD* et *ADXP* comme la figure 5.7 le montre.

Alors que le point précédent garantissait une cardinalité [0-n], il n'en est rien pour [1-n]. Pour garantir qu'il y ait au moins une valeur pour l'attribut, il est nécessaire de créer une contrainte supplémentaire d'équivalence entre les attributs concernés (les deux occurrences de *adID*). Malheureusement, une contrainte d'équivalence n'est possible que par l'ajout de deux clés étrangères de sens opposés entre les tables. Cela est embêtant pour les insertions car les contraintes vont les refuser. En effet, aucune référence n'est disponible dans la table correspondante. Pour cette raison, il a été décidé de réduire les [1-n] à [0-n]. Il s'agit d'une situation temporaire.

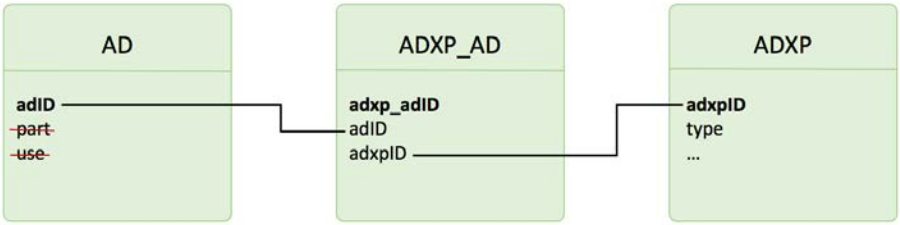

Figure 5.7 – Création d'une table intermédiaire pour la cardinalité [1-n]

En guise de résumé de cette quatrième étape de transformation, le code SQL des tables et des contraintes est présenté au code 5.7.

Code 5.7 – Code SQL après la quatrième étape pour *AD*

```
CREATE TABLE AD
(
   adID            Integer
);

CREATE TABLE POSTALADDRESSUSE
(
   ID              VARCHAR2(20 BYTE)       NOT NULL,
   LABEL           VARCHAR2(50 BYTE)       NOT NULL,
   DESCRIPTION     VARCHAR2(4000 BYTE)
);

CREATE TABLE ADXP
(
   adxpID          Number(19)              NOT NULL,
   type            VARCHAR2(10 BYTE)       NOT NULL,
   xpID            Number(19)
);

-- Tables intermédiaires
CREATE TABLE ADXP_AD
(
   adxp_adID       Number(19)              NOT NULL,
   adID            Number(19)              NOT NULL,
   adxpID          Number(19)              NOT NULL
);

CREATE TABLE USE_AD
(
   use_adID        Number(19)              NOT NULL,
   adID            Number(19)              NOT NULL,
   postalAddressUse varchar2(20 BYTE)      NOT NULL
);
```

```
34
35 -- Contraintes sur les tables intermédiaires
36 ALTER TABLE ADXP_AD ADD(
37 CONSTRAINT ADXP_AD_PK
38 PRIMARY KEY (adxp_adID)
39 USING INDEX ADXP_AD_PK
40 ENABLE VALIDATE) ;
41
42 ALTER TABLE USE_AD ADD(
43 CONSTRAINT USE_AD_PK
44 PRIMARY KEY (use_adID)
45 USING INDEX USE_AD_PK
46 ENABLE VALIDATE) ;
```

5. Remplacement des types de cardinalités [0-1]

La cardinalité [0-1] dénote un attribut facultatif. Pour cette cinquième étape, l'exemple de *ENXP*, représenté à la figure 5.8, est utilisé. La partie intéressante est l'attribut *type*, de cardinalité [0-1]. Un attribut facultatif est caractérisé, en SQL, par l'absence du mot-clé « not null ». Laisser le code inchangé n'est pas suffisant. En effet, les transformations précédentes obligent à réaliser une action supplémentaire. Cette action consiste à remplacer le type de données de l'attribut concerné par l'identifiant de la table nommée par cet attribut.

Figure 5.8 – Illustration du concept de *ENXP*

L'exemple du code 5.8 est sans doute plus clair. Enterprise Architect génère le code sans se préoccuper des cardinalités (A). Après les précédentes étapes de transformations, le code SQL devient celui présenté au point B. La méthodologie vise à éradiquer les types personnalisés. Il est nécessaire de faire une modification. Au point C, *EntityNamePartType* est remplacé par le type de l'identifiant de la table *EntityNamePartType*, qui est *varchar2*. Il est nécessaire d'appliquer une contrainte sur *type*. La clé étrangère qui réalise la contrainte est présentée au point D.

Code 5.8 – Illustration de la cinquième étape de transformation

```
1  -- A. Code SQL initial généré par Enterprise Architect
2
3  CREATE TYPE ENXP UNDER XP
4  (
5      type        EntityNamePartType,
6      qualifier   set_ENPQualifier,
7      enxpID      Integer
8  );
9
10
```

```
11 -- B. Code transformé par les étapes précédentes
12
13 CREATE TABLE ENXP
14 (
15     type        EntityNamePartType,
16 --  qualifier est supprimé : cardinalité 0-n
17 --  qualifier   EntityNamePartQualifier,
18     enxpID      Integer
19 );
20
21
22 -- C. Code SQL de la table (après étape 5)
23
24 CREATE TABLE ENXP
25 (
26     type        varchar2(20 BYTE),
27     enxpID      Integer
28 );
29
30 -- D. Code final de la contrainte
31
32 ALTER TABLE ENXP ADD(
33 CONSTRAINT ENXP_TYPE_ENPTYPE_FK
34 FOREIGN KEY (type)
35 REFERENCES EntityNamePartTYPE (ID)
36 ENABLE VALIDATE);
```

6. Traitement des relations *is-a*

Dans le monde informatique, il faut savoir qu'il existe quatre types de distribution concernant l'héritage [71] :

- le recouvrement avec couverture partielle (héritage classique) ;
- le recouvrement avec couverture totale ;
- la disjonction avec couverture partielle ;
- la disjonction avec couverture totale (partition).

La figure 5.9 illustre les différents cas possibles pour les relations *is-a*. Dans le modèle vMR, toutes les relations d'héritage font partie du premier cas : le recouvrement avec couverture partielle. Plusieurs méthodes existent [72] pour transformer les relations *is-a* en schéma relationnel et en code SQL. Cette transformation est nécessaire pour que le code soit conforme au SQL2. En effet, le code SQL2 n'accepte pas [71] :

- les attributs multivalués ;
- les attributs composés ;
- les types d'associations ;
- les relations is-a ;
- les types d'entités sans attribut ;
- les contraintes autres que celles des identifiants, des clés étrangères et d'existence ;
- les noms d'objets non conformes à la syntaxe SQL.

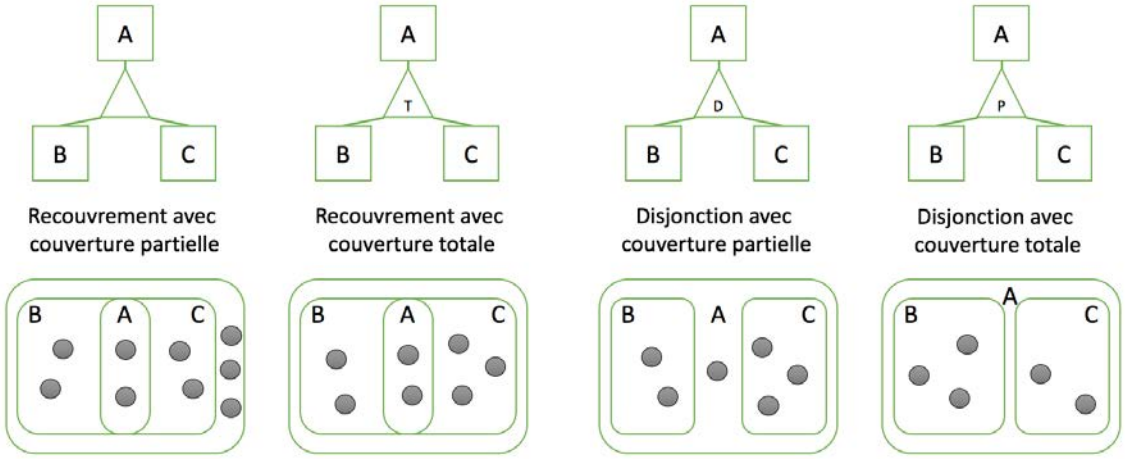

Figure 5.9 – Les différentes versions de l'héritage

Les méthodes pour transformer les relations *is-a* sont au nombre de 3 : la matérialisation, l'héritage descendant et l'héritage ascendant [72]. Comme il était exigé au départ de rester le plus près possible du standard vMR, la première méthode a été privilégiée. La matérialisation garde les tables « mères » et « filles » alors que dans les deux autres méthodes une table ou l'autre est supprimée, comme l'illustre la figure 5.10.

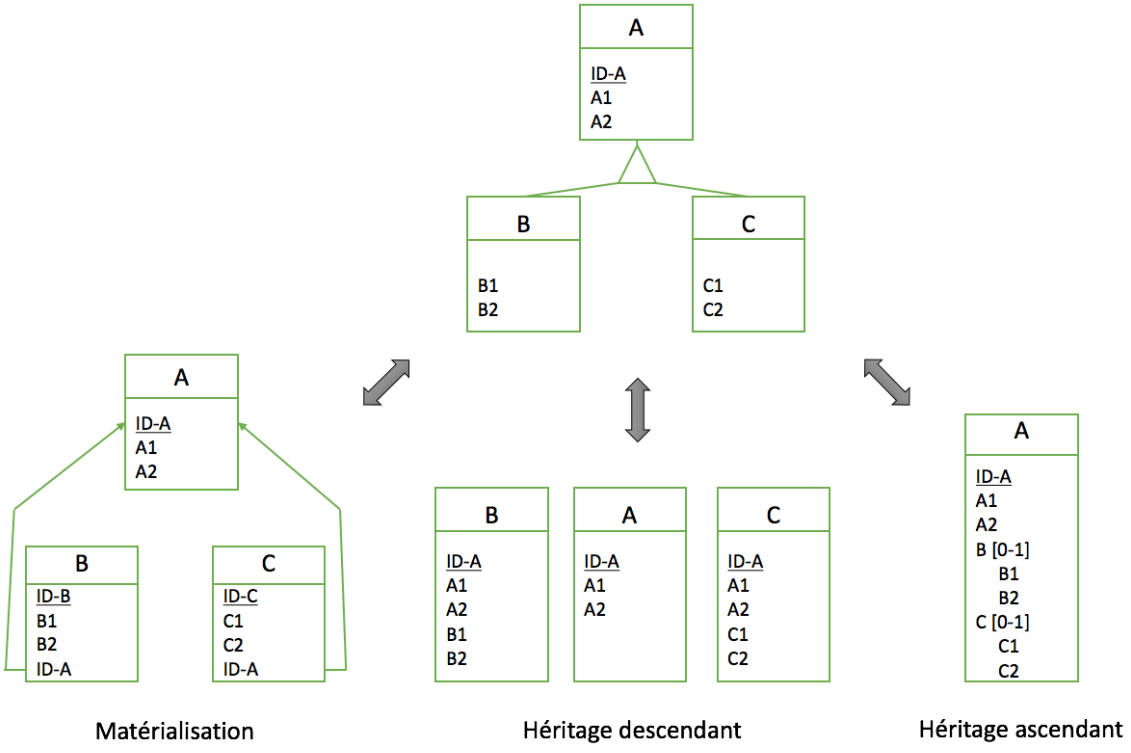

Figure 5.10 – Les trois techniques de suppression de l'héritage

Dans la matérialisation, il ne faut supprimer aucun élément : aucune table ni aucun attribut. Il faut néanmoins ajouter l'identifiant de la table « mère » dans les tables « filles ». Pour l'exemple de *XP*, cela revient à ajouter l'identifiant de *XP* dans *ADXP* et dans *ENXP*. Par la suite, une contrainte d'intégrité (clé étrangère) doit être créée entre les tables. Il paraît évident que l'identifiant ajouté doive faire référence à l'identifiant de la table « mère ». La figure 5.11 montre le passage d'une relation *is-a* au schéma conforme à la syntaxe SQL2.

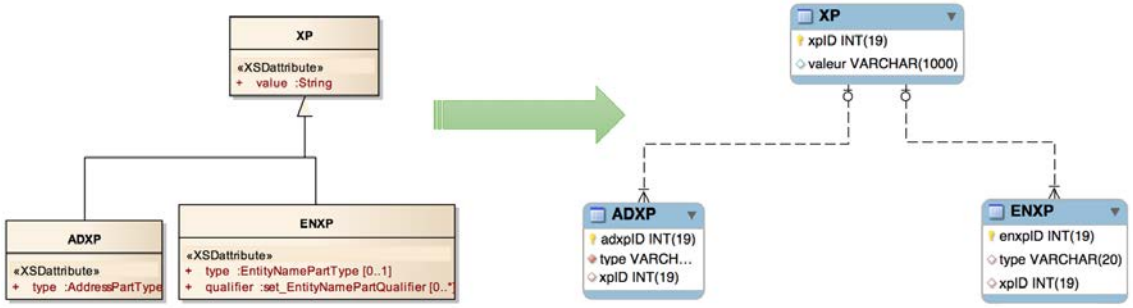

Figure 5.11 – Transformation de la relation *is-a*

7. Traitement des relations non *is-a*

Pour traiter les autres relations, il faut préciser que le modèle vMR possède 8 relations de navigabilité de deux types différents. En effet, la multiplicité associée à une terminaison d'association est soit [0-n] ou [1-1]. Le concept de navigabilité indique s'il est possible de traverser une association [73]. Dans le cas présent, la navigabilité n'est possible que dans un sens de par la représentation de la flèche.

Comme il en a déjà été question dans la quatrième phase de cette méthodologie, pour les relations de multiplicité [0-n], il est nécessaire d'introduire une table intermédiaire. Il faut alors inscrire les identifiants des deux tables correspondantes et ne pas oublier de réaliser les contraintes.

Pour les relations de multiplicité [1-1], l'entité éloignée de la terminaison d'association se voit recevoir l'identifiant de l'autre entité. En plus, comme le caractère facultatif n'est pas présent, le nouvel identifiant inscrit doit être unique. Pour cette raison, et en plus de la clé étrangère qui lie les deux tables, le nouvel identifiant doit être modifié en identifiant secondaire de sa table par le mot clé *unique*. L'illustration 5.12 expose ces deux cas.

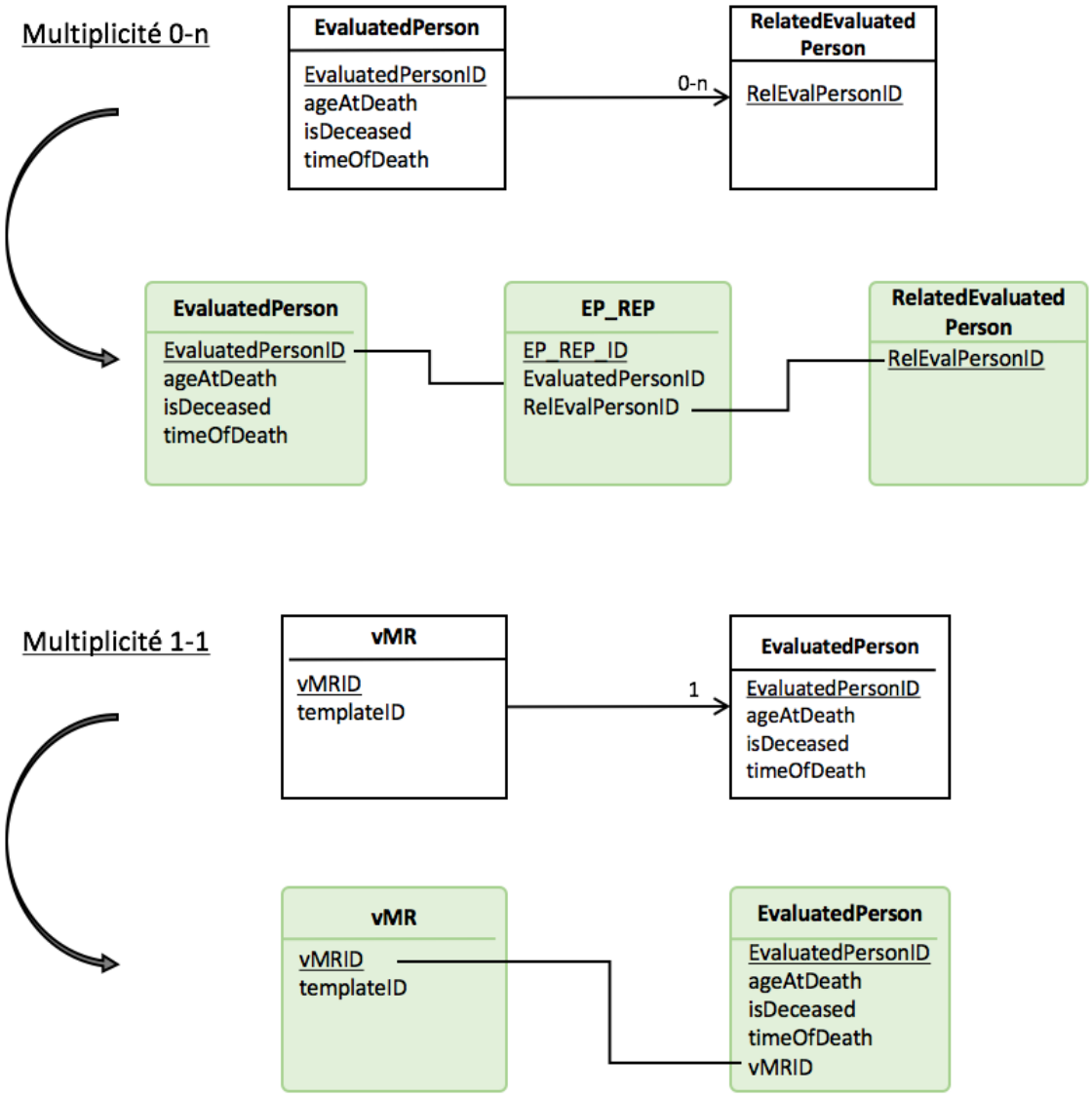

Figure 5.12 – Transformation des relations non *is-a*

8. Gestion des clés primaires et des index

Les grandes étapes de transformations sont maintenant terminées. Il reste encore à appliquer les clés primaires sur l'ensemble des tables et définir les index. Fort heureusement, les identifiants des tables ont été générés correctement par Enterprise Architect. Ces identifiants sont facilement repérables dans le code car ils sont nommés suivant cette structure : nom_de_la_table + « ID ». Il suffit alors d'ajouter le mot « not null » et la contrainte correspondante. Concernant les index, ils sont créés en fonction des identifiants. Le code 5.9 propose cette huitième étape de transformation.

Code 5.9 – Illustration de la huitième étape de transformation

```
-- Illustration avec l'exemple de AD

CREATE TABLE AD
(
    adID    Integer    NOT NULL,
    anyID   Integer
);

CREATE UNIQUE INDEX AD_PK
ON AD (adID)
LOGGING
TABLESPACE MIMS_INDEX
NOPARALLEL;

ALTER TABLE AD ADD (
CONSTRAINT AD_PK
PRIMARY KEY (adID)
USING INDEX AD_PK
ENABLE VALIDATE);
```

9. Transformation des types simples

Certains mots tels que *String* ou *Integer*, qui caractérisent les types simples, ne sont pas acceptés par les serveurs de bases de données. Il est alors conseillé de les transformer. Pour les chaînes de caractères, le mot-clé *varchar2* a été choisi tandis que pour les entiers, il s'agit de *number(19)*.

Code 5.10 – Illustration de la neuvième étape de transformation

```
-- Illustration avec l'exemple de AD

CREATE TABLE AD
(
    adID    Number(19)    NOT NULL,
    anyID   Number(19)
);
```

Quid de la récursivité ?

Dans les problèmes potentiels exposés à la section 5.3.1, il a été question de la récursivité des types de données, notamment avec le concept de *CD*. Après les transformations réalisées à l'aide de la méthodologie, la récursivité a disparu. En effet, la création des tables supplémentaires a permis de l'éradiquer. A la place, les identifiants sont enregistrés dans la table faisant office d'intermédiaire.

5.4 Code SQL Corrigé et Schéma Relationnel

Pour conclure ce chapitre, cette section vise à présenter les schémas finaux du modèle vMR, qui représentent en fait le schéma relationnel complet de la base de données, ainsi que le code SQL transformé via la méthodologie décrite précédemment.

La figure 5.13 résume le cycle de transformation. Le modèle vMR a été choisi comme point de départ pour la transformation. La première étape est de générer le code SQL pour commencer à utiliser la méthodologie. Ensuite, les neufs étapes de transformation décrites dans la méthodologie permettent d'arriver au code SQL final. Enfin, le logiciel MySQLWorkBench [74] a été utilisé pour générer les schémas.

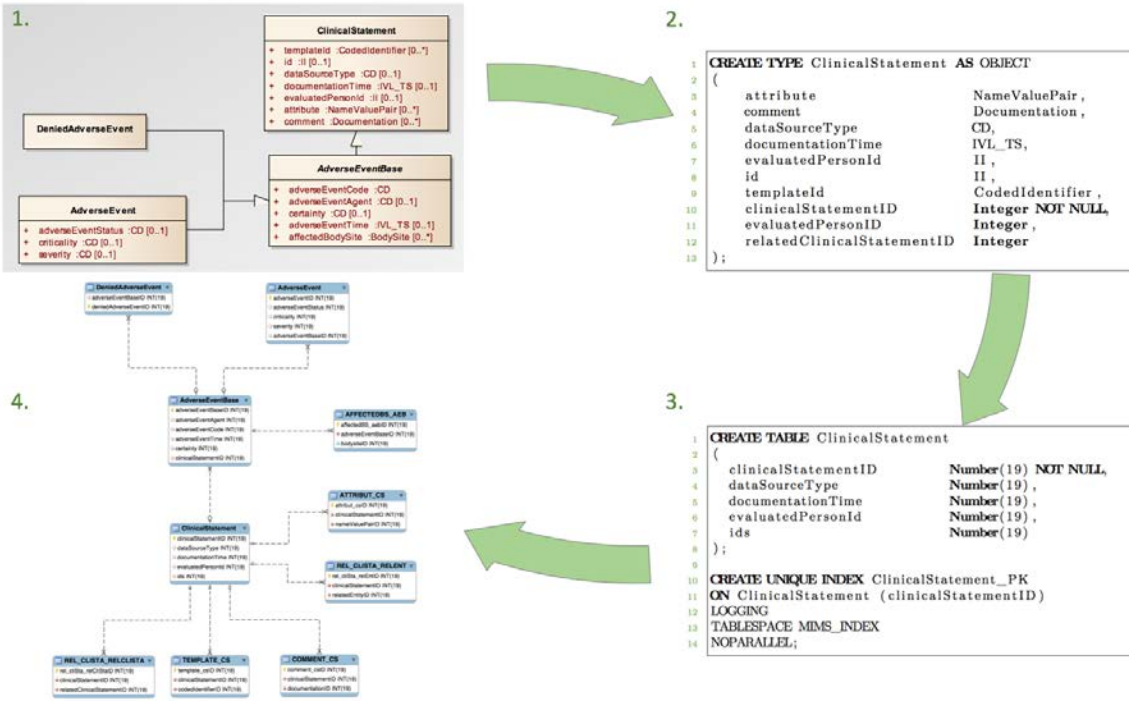

Figure 5.13 – Transformation du modèle vMR vers les schémas relationnels

En guise de complément, le lecteur est invité à se diriger vers les annexes B et C. L'annexe B regroupe les différents diagrammes initiaux de vMR tandis que l'annexe C présente les schémas relationnels correspondant aux 20 concepts du milieu médical repérés par le modèle vMR.

Chapitre 6

Exploration des Données

Ce chapitre traite de l'exploration de données (i.e. *data mining*) résultant des requêtes introduites par les utilisateurs dans les formulaires dynamiques de l'application web. Après avoir réalisé une approche théorique du data mining au chapitre 3, ce chapitre-ci traite plutôt de la façon dont les méthodes et algorithmes existants ont été intégrés dans le code de l'application RetroStudy.

6.1 Choix de l'Outil

Chacun des logiciels présentés au chapitre 3, Weka et R, sont équivalents en ce qui concerne leurs fonctionnalités. Des tests ont été réalisés sur chacun d'eux pour pouvoir choisir efficacement lequel des deux serait le plus adapté à RetroStudy. Tout d'abord, les tests se sont portés sur la version standalone. La simplicité des deux logiciels ne permettait pas de trancher. Chacun d'eux pouvait aisément réaliser les tâches de data mining demandées ainsi que produire des graphiques.

Après avoir testé les logiciels, il fallait être certain que leur utilisation dans un projet informatique était possible. Malheureusement, R n'offrait pas de librairie pouvant être intégrée directement dans un environnement de programmation (Eclipse). Bien qu'une intégration partielle de R ait été établie, les valeurs calculées ne pouvaient être récupérées dans le code Java de RetroStudy. Le choix du logiciel s'est donc porté vers Weka. Toutes les méthodes et tous les algorithmes sont disponibles dans la librairie correspondante. Par contre, en ce qui concerne les graphiques, la librairie Weka ne permet pas, en tout cas directement, la création d'images illustrant les graphiques.

6.2 Algorithmes Utilisés

La sélection des algorithmes de data mining s'est déroulée lors d'une entrevue avec le professeur B. Frénay, expert en data mining et machine learning à l'Université de Namur. Lors de cette réunion, le contexte de l'application web et le temps imparti pour la réaliser ont été clairement exposés. Selon ses dires, il n'était pas nécessaire de réinventé la roue et d'implémenter une nouvelle fois les algorithmes d'exploration de données. Comme le temps d'implémentation était particulièrement court, il fallait trouver un compromis entre une application fonctionnelle et une application complète. Dans ce sens, le Pr. Frénay a conseillé de débuter l'implémentation de RetroStudy par les méthodes de classification.

Comme il l'a déjà été signalé auparavant, les méthodes de classification regroupent un certain nombre d'algorithmes allant des simples arbres de décision aux complexes réseaux de neurones. Toujours selon le Pr. Frénay, il était évident de s'orienter dans un premier temps vers les arbres de décision et d'enrichir progressivement l'application web avec d'autres méthodes plus complexes. De plus, ils ont l'avantage d'offrir une visualisation basique de l'information sous forme d'arbres. C'est donc tout naturellement que les arbres de décision ont été choisis.

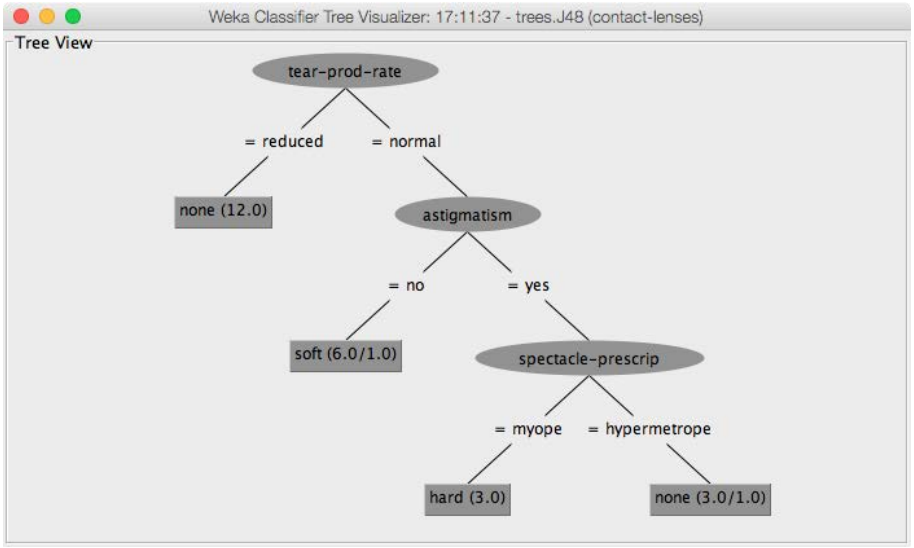

Figure 6.1 – Illustration d'un arbre de décision dans Weka

Les arbres de décision font donc partie des méthodes de classification et plus globalement des techniques supervisées (ou prédictives). C'est pour classer une population de données en différents groupes homogènes selon un objectif fixé au départ que cette technique est prisée actuellement. En effet, les arbres de décision sont un excellent moyen pour visualiser l'information clé dans un immense amas de données. Leurs caractéristiques sont :

— **la lisibilité** La représentation des données se fait sous la forme d'un arbre facile à comprendre. Un schéma clair est plus expressif quand il s'agit de synthétiser une multitude d'éléments.
— **le caractère automatique** Les algorithmes d'arbres de décision sont capables de sélectionner automatiquement les données qui leur semblent intéressantes (variables dites discriminantes) parmi un large choix de données potentielles. Toutes les variables ne sont pas forcément retenues.
— **la capacité de traitement** Les arbres de décision sont un bon moyen pour appréhender les gros fichiers de données et y faire ressortir l'information essentielle.
— **la rapidité d'exécution** Les algorithmes sont rapides car peu d'hypothèses sont faites au départ.

Au sein de la famille des arbres de décision, plusieurs algorithmes sont possibles et de certaines distinctions apparaissent suivant celui utilisé. Les algorithmes possibles sont par exemple J48, Arbres aléatoires, Forêts aléatoires, etc. En fonction de l'algorithme choisi, il est nécessaire de discrétiser les données. « *Discrétiser une variable quantitative c'est, mathématiquement, transformer un vecteur de nombres réels en un vecteur de nombres entiers nommés indices de classe* » [75]. Concrètement, pour une variable donnée, discrétiser la variable serait de découper les valeurs en plusieurs groupes. Dans le cas d'une variable « taille » par exemple, qui mesure la taille en cm des individus, il serait envisageable de créer deux groupes : les grands (plus de 150 cm) et les petits (moins de 150 cm) [1]. L'algorithme de data mining travaille alors sur le nom des regroupements et plus sur les données numériques.

Concernant RetroStudy, il a été choisi d'implémenter dans un premier temps l'algorithme d'*Arbre aléatoire* (*Random Tree*) qui ne requiert pas la discrétisation des données. Dans le futur, il serait intéressant d'intégrer d'autres formes d'algorithmes de data mining dont celles d'arbres de décision.

6.3 Intégration dans RetroStudy

La librairie Weka a été directement intégrée au projet RetroStudy. Tous les algorithmes pouvaient alors être utilisés à l'aide de méthodes Java. Dans cette section sera présentée la manière dont l'intégration s'est déroulée mais également les modifications qui ont dû avoir lieu pour permettre une exécution correcte du code.

6.3.1 Préparation des Données

La préparation des données est imposée par le data mining sous l'étape de prépocessing. Pour rappel, cette étape pouvait être composée de quatre phases distinctes : le nettoyage des données, l'intégration des données, la sélection des données et la transformation des données. L'étape du préprocessing ne doit pas être négligée car sans elle les algorithmes de data mining risquent de ne pas pas s'exécuter correctement.

1. Cette décomposition a été choisie de façon tout à fait arbitraire.

Nettoyage des données En procédant étape par étape, il faut d'abord nettoyer les données en supprimant les données inconsistantes et les doublons. Comme les données contenues dans la base de données IFLEC sont pour la plupart des données générées, cette première phase est inutile.

Intégration des données Cette deuxième phase n'est également pas nécessaire dans le cas de notre application web. En effet, le projet IFLEC ne possède pas de sources multiples en ce qui concerne les données. L'intégration n'a donc pas lieu.

Sélection des données En guise de sélection des données, il faut restreindre l'échantillon de départ aux données qui semblent les plus intéressantes. Pour cela, il a été décidé de choisir toutes les données qui sont liées entre elles. Par exemple, lorsque l'utilisateur réalise une requête via RetroStudy, les catégories de contraintes que l'utilisateur a remplies sont retenues par l'application. Lorsqu'arrive le moment de procéder à l'exploration de données, l'application récupère toutes les informations contenues dans les catégories sur les patients concernés [2].

A ce moment-ci de l'exploration des données, ces dernières sont brutes. Ceci veut dire qu'aucun traitement n'a eu lieu vis-à-vis des données ni aucun fichier n'a été créé. Les données se trouvent encore dans la mémoire virtuelle de la machine. La phase suivante modifie les données pour les faire correspondre à la méthode de data mining choisie.

Transformation des données Faire correspondre les données récupérées de la base de données avec les valeurs autorisées n'est pas une tâche aisée. Chaque catégorie d'algorithmes et parfois même chaque algorithme n'autorise pas les mêmes valeurs qu'un(e) autre. Dans le cas des arbres aléatoires, les valeurs textuelles ne sont pas accordées par l'algorithme. Au risque de provoquer une erreur, il faut modifier les données. Par chance, les données qui sont enregistrées dans la base de données et qui peuvent être utilisées pour de l'exploration de données sont majoritairement chiffrées. Seuls le genre et la date de naissance sont enregistrés sous forme textuelle. Dans le cas du genre, le patient de sexe masculin est encodé en « 0 » et le patient de sexe féminin est encodé en « 1 ». Pour la date de naissance, l'âge de la personne est calculé. Ainsi, toutes les données sont maintenant sous forme numérique.

Le logiciel et la librairie Weka ont besoin d'un fichier avec une extension un peu particulière. Il s'agit d'un fichier .arff (*Attribute-Relation File Format*). Ce type d'extension est propre à Weka et a également été développée par l'Université de Waikato en Nouvelle-Zélande [59]. La structuration d'un tel fichier est simple et ne comporte que trois sections : le nom de la relation, les attributs et les données. Un schéma de la structure du fichier .arff est proposée à la figure 6.2.

2. Le lecteur est invité à se référer au chapitre 7 qui traite des interfaces utilisateur. La fonctionnalité de création de contraintes via un formulaire dynamique y est expliqué plus en détail.

Figure 6.2 – Schéma de la structure d'un fichier .arff

Au niveau du nom de la relation, il s'agit simplement de fournir à titre indicatif le nom de la relation qui vise à être découverte par l'algorithme de data mining. Cette information est contenue au niveau du *header* dans le fichier et ne possède pas de rôle important. Concernant les attributs, il faut lister les variables qui seront manipulées lors de la phase d'exploration. La manière de lister les éléments est de commencer par « @attribute » suivi du nom de la variable ainsi que de son type (numérique, textuel, etc.). Il est aussi possible de définir des classes, c'est-à-dire un attribut qui regroupe une liste de données textuelles. Par exemple, un attribut « production_de_larmes » pourrait contenir « normal » ou « faible ». Concernant les données, elles sont simplement, pour chaque individu, précédées de « @data », inscrites dans l'ordre des attributs et séparées par une virgule. Un exemple illustratif est présent au code 6.1 pour faciliter la compréhension.

Code 6.1 – Exemple d'un fichier .arff

```
@relation contact-lenses

@attribute age                  {young, pre-presbyopic, presbyopic}
@attribute spectacle-prescrip   {myope, hypermetrope}
@attribute astigmatism          {no, yes}
@attribute tear-prod-rate       {reduced, normal}
@attribute contact-lenses       {soft, hard, none}

@data
young,myope,no,reduced,none
young,myope,no,normal,soft
young,myope,yes,reduced,none
young,myope,yes,normal,hard
young,hypermetrope,no,reduced,none
young,hypermetrope,no,normal,soft
young,hypermetrope,yes,reduced,none
...
```

Sur l'exemple ci-dessus, tiré des fichiers de Weka, les données ont été discrétisées. Dans l'application web RetroStudy, elles ne l'ont pas été. Pour imaginer un fichier .arff tel qu'il l'est dans RetroStudy, il faut remplacer les catégories entre accolades par le type de l'attribut en question. Dans le cas présent, il s'agira systématiquement de variables numériques.

6.3.2 Exploration des Données

L'exploration des données consiste à exécuter l'algorithme choisi auparavant. Dans le cas des arbres aléatoires, l'exécution se déroule assez rapidement a priori. Le traitement s'effectue sur le fichier .arff créé dans les étapes précédentes. Il est tout de même intéressant de se pencher sur le temps d'exécution de l'algorithme en fonction du nombre d'instances contenues dans le fichier .arff. La figure 6.3 tente de faire le point sur le sujet. Les mesures ont été réalisées sur un MacBook Pro de 2011 avec un processeur Intel i5 double cœur de 2.3 GHz et 8 Go de RAM.

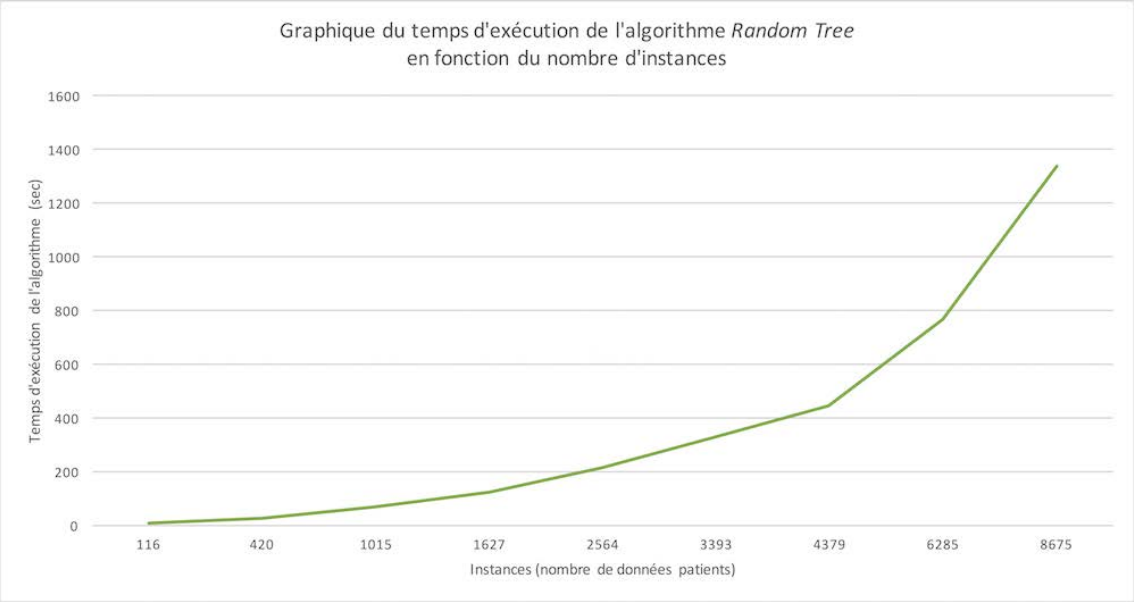

Figure 6.3 – Temps d'exécution de l'algorithme *Random Tree*

6.3.3 Visualisation des Données

La visualisation des résultats est une étape importante car c'est elle qui permet toute la compréhension de l'information. Beaucoup plus expressive que du texte, une illustration est incontestablement nécessaire dans l'application web. Une représentation schématique des données par un graphe et par un arbre a été choisie pour ce cas précis de data mining. En effet, chaque catégorie d'algorithmes possède sa propre représentation des données. Par exemple, dans le cas d'une description d'un concept, un diagramme en bâtons est amplement suffisant.

Afin de permettre un affichage du graphe dans RetroStudy, un programme ou une librairie supplémentaire devait être intégrée au projet web. Un petit programme, nommé GraphViz, disponible aussi sous forme de librairie, peut générer des graphes et même des arbres dans une qualité supérieure et dans de nombreux formats d'exportation. Comme tous les programmes de transformation de données, des données en entrée sont requises. Pour ce faire, un interfaçage avec l'outil Weka semblait possible au départ. En y regardant de plus près, le fichier requis par GraphViz devait porter l'extension .dot. C'est donc ce fichier - et heureusement, Weka peut réaliser une exportation des données dans ce format - qui est utilisé par GraphViz.

Concrètement, afin de visualiser l'arbre de décision à l'écran, l'outil Weka réalise une exportation des données vers un fichier temporaire .dot. Ce fichier est utilisé au niveau du code source par une méthode Java qui génère une image du graphe à l'aide de la librairie GraphViz. L'image est ensuite exportée en .pdf et est stockée au niveau du serveur Windows mis à disposition pour le projet. Garder les images évite de réaliser la transformation à chaque fois que l'utilisateur accède au site web. Une standardisation dans le nommage des images permet facilement de retrouver celle qui correspond à l'utilisateur.

La figure 6.4 montre l'exemple des lentilles de contact (figure 6.1) transformé à l'aide de GraphViz. L'image ainsi créée est beaucoup plus claire et facile à comprendre que des données textuelles. La racine de l'arbre ainsi que les nœuds sont illustrés par un ovale blanc et les feuilles par un rectangle gris.

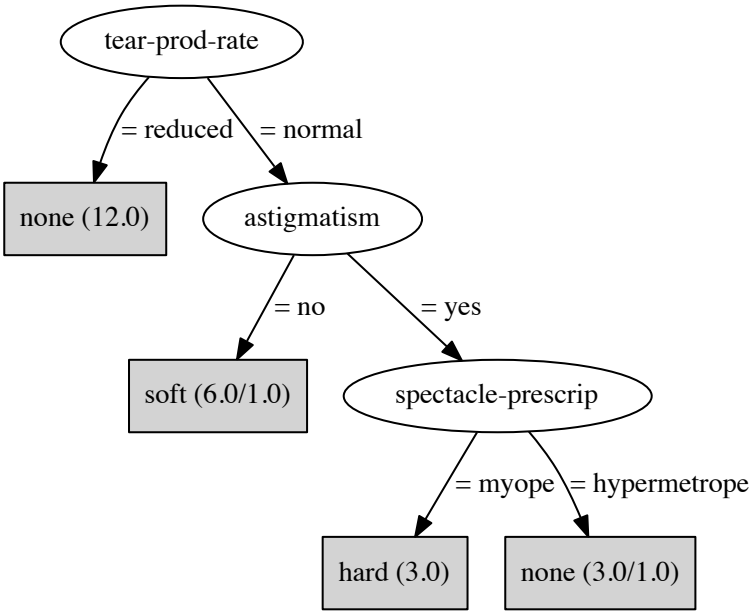

Figure 6.4 – Illustration d'un arbre de décision dans RetroStudy

D'autres informations viennent s'ajouter à la représentation de l'arbre de décision. Deux catégories d'informations y figurent : les données présentes sur les arbres de décision et les données textuelles générées par Weka.

Premièrement, en ce qui concerne les annotations sur les feuilles de l'arbre, des informations chiffrées sont disponibles entre parenthèses. Dans le cas où une seule valeur est représentée, il s'agit du nombre d'instances correctement classées (poids des instances) dans la feuille. Lorsque deux valeurs sont affichées et séparées par une virgule, la deuxième valeur représente les instances qui sont mal classées par l'algorithme. La première valeur est similaire au premier cas. Concrètement, pour la feuille « soft (6.0,1.0) », cela signifie que 6 instances ont été classées dans cette catégorie dont 1 qui était mal classée. Lorsqu'il s'agit de nombres avec des décimales, cela signifie que des données étaient manquantes et que l'algorithme a dû s'adapter.

Deuxièmement, un résumé de l'opération de data mining est généré par l'algorithme. Il permet de donner un aperçu un peu plus détaillé que l'arbre en lui-même. Les valeurs qu'il génère sont essentiellement des calculs de taux d'erreurs réalisés à l'aide de méthodes différentes. Le code 6.2 présente les informations supplémentaires générées par Weka.

Code 6.2 – Représentation des informations supplémentaires générées par Weka

```
=== Summary ===

Size of the tree : 19
Time taken to build model: 0 seconds
Correctly Classified Instances          17               70.8333 %
Incorrectly Classified Instances         7               29.1667 %
Kappa statistic                          0.4581
Mean absolute error                      0.2083
Root mean squared error                  0.433
Relative absolute error                 55.1471 %
Root relative squared error             99.1456 %
Coverage of cases (0.95 level)          75      %
Mean rel. region size (0.95 level)      37.5    %
Total Number of Instances               24
```

Pour compléter la visualisation des données, un autre système a été développé pour les médecins. Ce système consiste à manipuler dynamiquement des données sur un graphe. Cette fonctionnalité supplémentaire pourrait faire apparaître certains patterns à l'écran. La fonctionnalité en question est détaillée dans le chapitre suivant.

Chapitre 7

Interactions avec l'Utilisateur

En guise de présentation du logiciel RetroStudy, ce chapitre vise à donner un compte rendu détaillé des fonctionnalités et des interfaces graphiques du site web. A travers ces pages, il est question du cheminement typique que pourrait réaliser un utilisateur lambda sur l'application RetroStudy. A cette fin, le parcours commencera par la connexion au site et se terminera par la déconnexion de l'utilisateur.

7.1 Connexion

La partie sécurité est quelque chose qui ne devait pas être laissé de côté lors de l'implémentation. Celle-ci se décompose en deux parties : le chiffrement des mots de passe des utilisateurs et l'envoi sécurisé des données via internet.

Il a été décidé d'utiliser la méthode SHA-1 pour hasher le mot de passe et produire un texte chiffré. Ce texte est stocké dans la base de données au niveau de l'utilisateur. Au moment de se connecter au site web, l'utilisateur entre son identifiant et son mot de passe. Une méthode Java réalise une manipulation des données de l'utilisateur (identifiant et mot de passe) avec l'algorithme SHA-1. Le texte chiffré produit par cette méthode est comparé avec celui se trouvant en base de données. Si les données correspondent, alors l'utilisateur a accès au site web.

Afin de garantir une navigation sécurisée, il fallait absolument travailler avec le protocole HTTPS. Ce protocole est une variante de HTTP et fournit une communication chiffrée entre un utilisateur et un serveur web. Il s'agit en réalité du protocole HTTP auquel une couche de chiffrement SSL a été ajoutée [76]. Pour garantir la

sécurité des données, le site web fournit à l'utilisateur un certificat émis par une autorité tierce fiable. Dans le cas présent, il s'agit néanmoins d'un certificat auto-signé, c'est-à-dire un certificat émis par le créateur du site web. Cela n'empêche nullement le chiffrement des données.

Pour illustrer la fonctionnalité, l'écran de connexion est présenté à la figure 7.1. Sur cet écran, l'utilisateur est invité à entrer son identifiant personnel et son mot de passe. Le cadenas dans l'URL garantit l'envoi sécurisé des données.

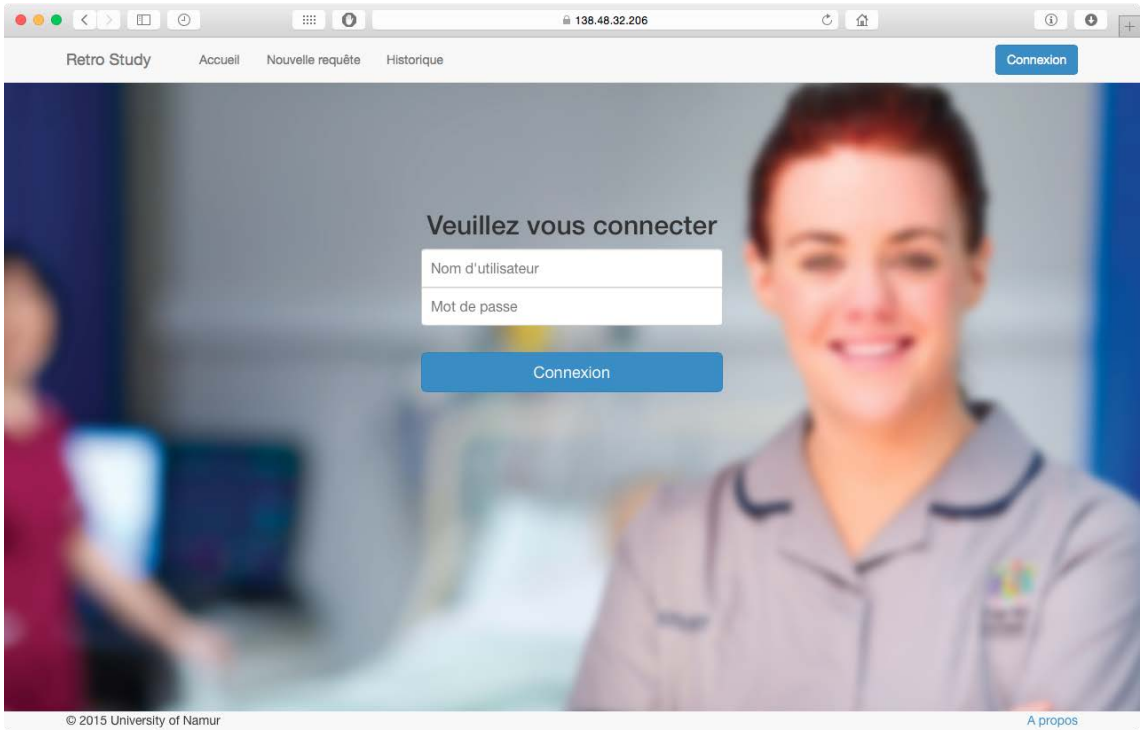

Figure 7.1 – Écran de connexion à l'application web

7.2 Formulaire Dynamique

Le système de requête est la fonctionnalité la plus complexe du système. Elle autorise l'utilisateur à créer lui-même une contrainte plus ou moins complexe sur les patients.

Une fois la page chargée, elle offre la possibilité de choisir la catégorie sur laquelle l'utilisateur désire faire une contrainte. Par défaut, il s'agit des données sur le patient. Dans cette catégorie, plusieurs éléments peuvent s'y retrouver. Par exemple, pour la catégorie *Données du patient*, des contraintes sur le sexe ou sur l'âge du patient sont disponibles. L'utilisateur peut ensuite introduire dans les deux champs suivants les valeurs minimales et maximales qu'il souhaite voir figurer dans la contrainte. Une autre possibilité est de n'introduire que la valeur minimale.

Lorsque c'est nécessaire, l'utilisateur peut choisir d'ajouter des contraintes dans une catégorie ou alors d'ajouter une autre catégorie. Pour cela, deux variétés de boutons sont placés dans le formulaire : un bouton vert « + » et un bouton « ajouter une catégorie ». Le premier ajoute une ligne dans la catégorie courante. Le second ajoute une nouvelle catégorie de contraintes à la fin du formulaire. De cette manière, le formulaire préserve le côté dynamique dans la création des requêtes.

Pour illustrer le fonctionnement du formulaire, reprenons la requête d'un médecin déjà présentée au chapitre 4 :

Quels sont les patients de sexe masculin de moins de 40 ans et de plus de 18 ans ayant une glycémie élevée ?

Dans cette requête, trois contraintes sont présentes : le sexe, l'âge et le niveau de glycémie. Pour réaliser cette requête sur l'application web, il faut premièrement choisir le type de catégorie. Pour effectuer l'exemple dans l'ordre de la requête, il faut d'abord choisir *Données du patient*. Là, les options disponibles pour les contraintes apparaissent. Il faut ensuite sélectionner *Sexe* dans la liste déroulante et y inscrire la valeur *M* dans le premier champ. Cela correspond à choisir tous les patients de sexe masculin. Pour continuer la requête, il faut cliquer sur le bouton vert « + » qui ajoute alors une ligne dans la catégorie. Il faut maintenant choisir *Âge* dans cette nouvelle liste déroulante et indiquer les valeurs *18* et *40* dans les deux champs suivants. Pour la dernière contrainte, il faut ajouter une catégorie et sélectionner *Biologie clinique*. L'utilisateur peut maintenant choisir dans la liste déroulante suivante *Glycémie* et indiquer les valeurs correspondantes [1].

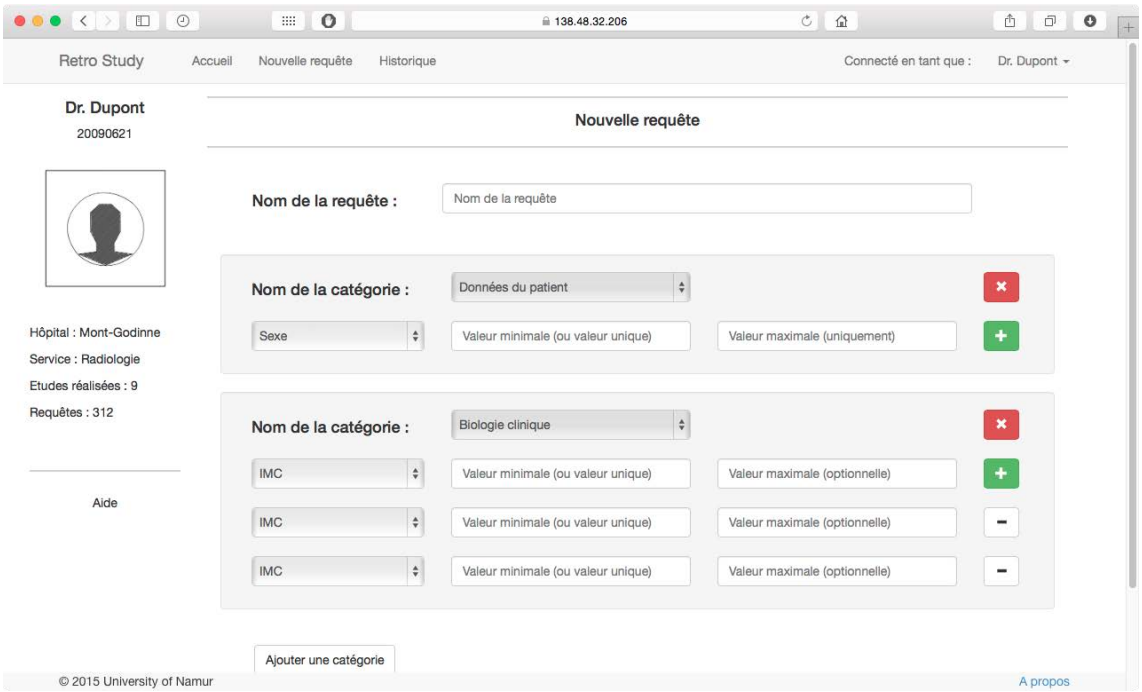

Figure 7.2 – Écran présentant le formulaire dynamique

1. La glycémie se mesure en mg/dl de sang et la norme se situe entre 80 et 120. Une glycémie élevée correspond donc à plus de 120 mg/dl [77].

La figure 7.2 montre le formulaire dynamique où une deuxième catégorie a été ajoutée ainsi que des contraintes dans celle-ci.

Afin de proposer un tel formulaire aux utilisateurs, il fallait utiliser un autre langage que le HTML pour élaborer cette page web. Ce langage est le Javascript et définit des fonctions qui s'exécutent sur le navigateur de l'utilisateur. Il permet de faire des traitements parfois lourds sans encombrer le serveur inutilement. Les fonctions qui ont été créées permettent d'ajouter des catégories au formulaire mais aussi des contraintes dans ces catégories.

7.3 Historique des Requêtes

Chaque requête émise par l'utilisateur est enregistrée dans un fichier XML reprenant les paramètres de la requête, les patients correspondants, les valeurs chiffrées et les statistiques de data mining. Comme les fichiers sont enregistrés sur le serveur, un système d'historique peut être mis en place, ce qui a été fait. Une page web spécifique a été créée (figure 7.3) et permet de visualiser les requêtes précédentes. Lorsque l'utilisateur sélectionne une requête de la liste, celui-ci a accès à l'ensemble des résultats (voir section 7.4).

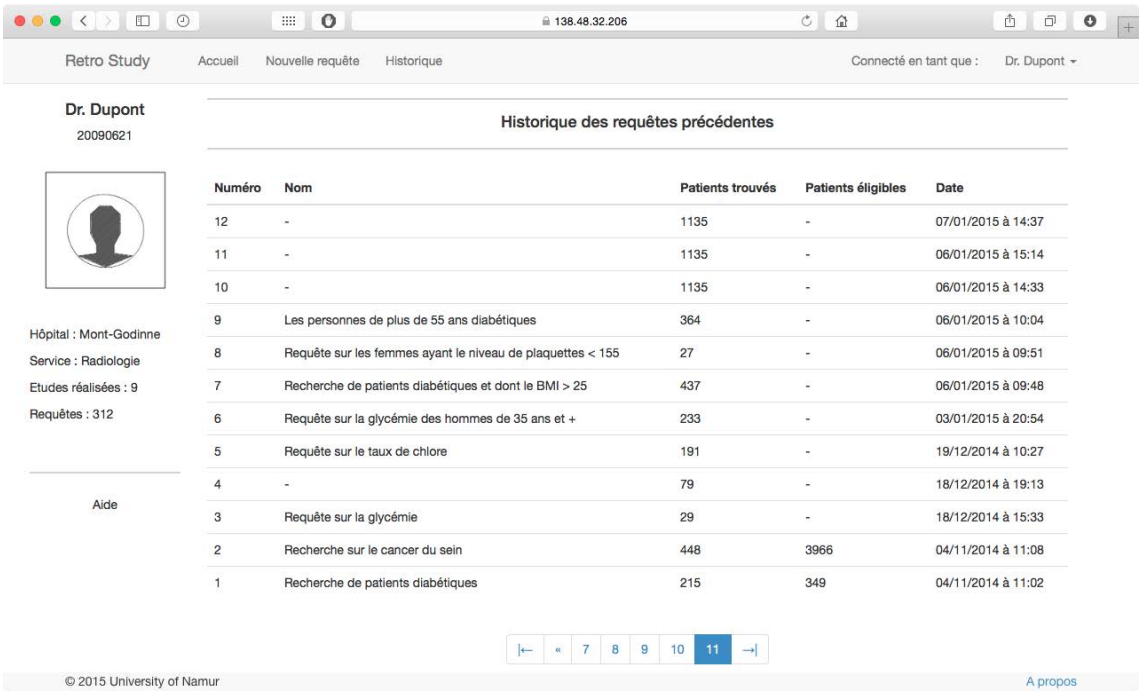

Figure 7.3 – Écran d'historique des requêtes

7.4 Résultats

Au moment de soumettre le formulaire au serveur, l'utilisateur est invité à choisir la méthode d'exploration de données. Ceci étant fait, une page web présente les résultats du traitement de la requête. Ceux-ci sont répartis suivant plusieurs catégories : la liste des patients, la visualisation des données sur un graphe, les statistiques de data mining et la liste des contraintes. Les sous-sections suivantes détaillent ces catégories.

7.4.1 Liste des Patients

La première catégorie est l'affichage des patients. Ceux-ci sont classés dans un tableau reprenant leur nom, prénom, date de naissance et numéro de téléphone. Ils représentent les patients éligibles pour une nouvelle étude clinique. Une étude clinique possède des critères d'entrée qui sont représentés par les contraintes émises lors de la requête par l'utilisateur. Comme le but est de trouver de nouveaux patients pour l'intégration dans les études cliniques, une fonctionnalité d'ajout de patients dans une étude a été envisagée. Pour le moment, seule une case à cocher est disponible lors de l'affichage des patients et devrait à terme permettre à l'utilisateur de confirmer l'ajout d'un patient à cette étude.

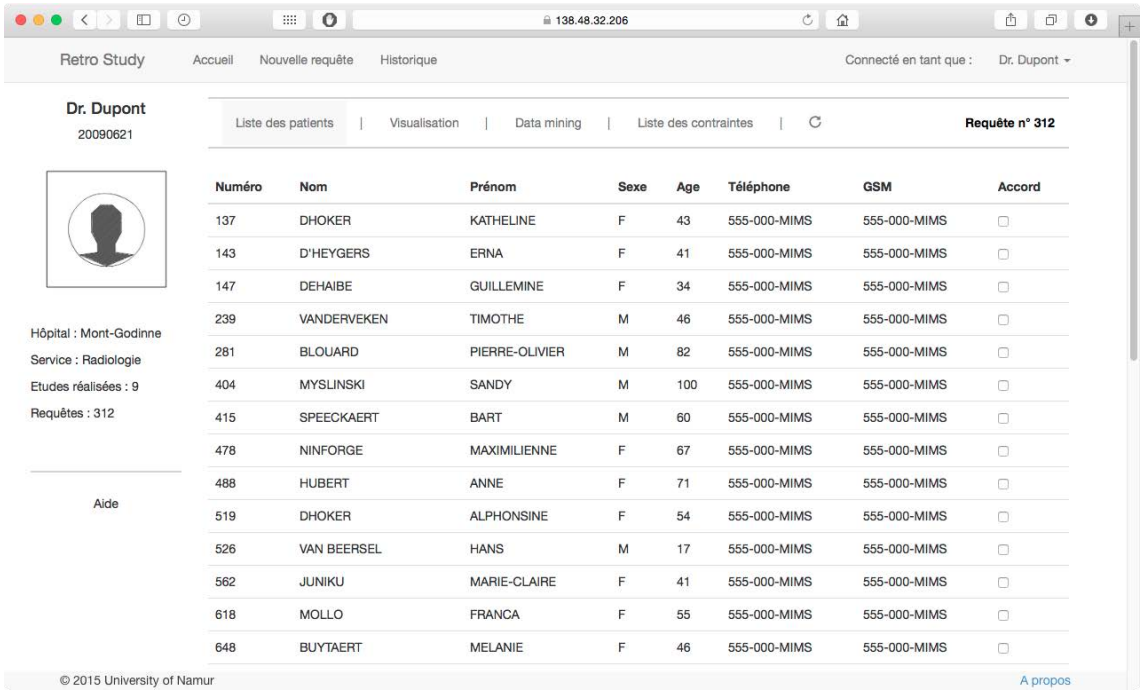

Figure 7.4 – Écran présentant la liste des patients éligibles pour une étude clinique

7.4.2 Visualisation des Résultats

La deuxième catégorie de résultats est la visualisation des données sur un graphe. Lorsque l'utilisateur réalise une requête, les catégories de contraintes sont mémorisées et les valeurs chiffrées de toutes les catégories sélectionnées sont automatiquement disponibles sur un graphe. Ainsi, même si une requête ne concerne que la glycémie du patient, le taux de globules blancs est également disponible.

L'utilisateur a la possibilité de choisir les valeurs qu'il souhaite voir apparaître sur l'axe des ordonnées et sur l'axe des abscisses. De cette manière, c'est lui qui décide des meilleures valeurs à comparer. Une fois qu'il a choisi les variables dans les listes déroulantes appropriées, il peut valider ses choix en cliquant sur le bouton « Exécuter ». Le graphique se transforme alors de lui-même. Dans cette première version de l'application, les valeurs sont groupées selon le sexe. Il faut imaginer une amélioration à cette fonctionnalité où n'importe quelle variable pourrait faire office de variable de regroupement. Cela pourrait se faire notamment via des techniques de data mining avancées.

La figure 7.5 illustre la visualisation des données sur un graphique. Il a été choisi de comparer le taux de calcium avec l'IMC (indice de masse corporelle). L'utilisateur peut ou non remarquer des regroupements de valeurs suivant les variables choisies dans les axes. Les données d'illustration sont des données générées aléatoirement, suivant une distribution uniforme. Ceci explique la diversité des valeurs et surtout le fait de ne pas remarquer des regroupements de données à l'écran.

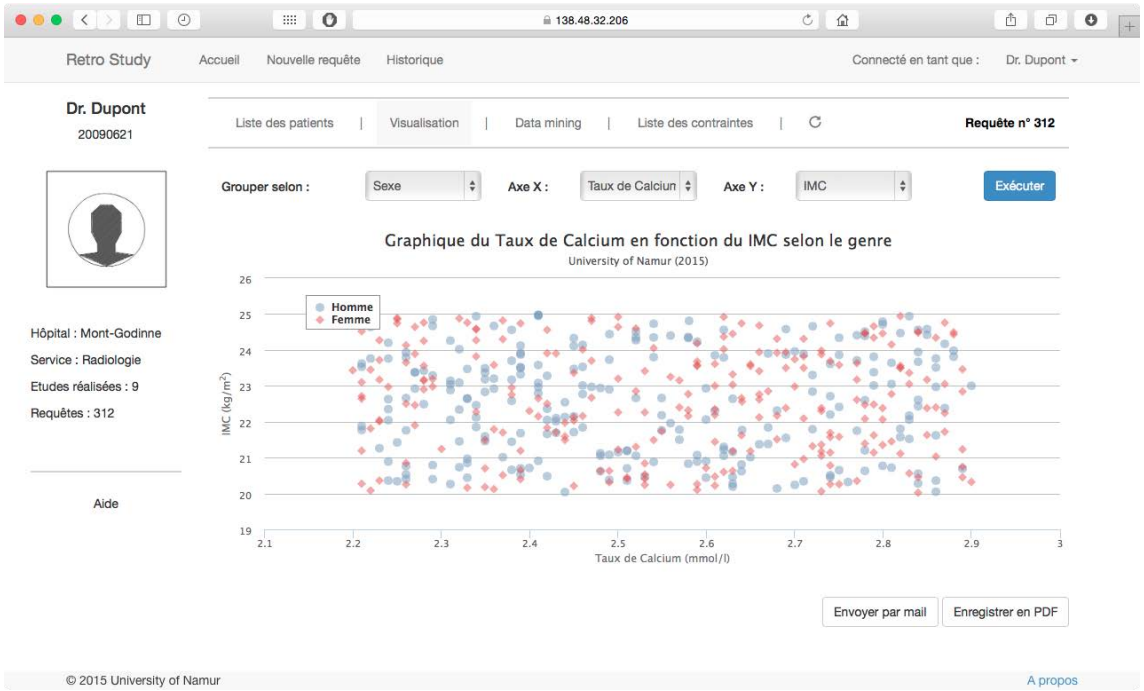

Figure 7.5 – Écran de visualisation des résultats sur un graphe

7.4.3 Statistiques de Data Mining

Les fonctionnalités de data mining qui ont été présentées dans le chapitre 6 sont utilisées pour produire des statistiques sur les données chiffrées ainsi qu'une représentation graphique de ces données, plus facilement compréhensible par le médecin. La représentation graphique peut aider le médecin à trouver quelle variable a plus de poids dans la requête. Elle donne un indicateur sur la classification des valeurs suivant les individus concernés. La figure 7.6 expose les statistiques et l'arbre généré (représentation graphique) pour une requête.

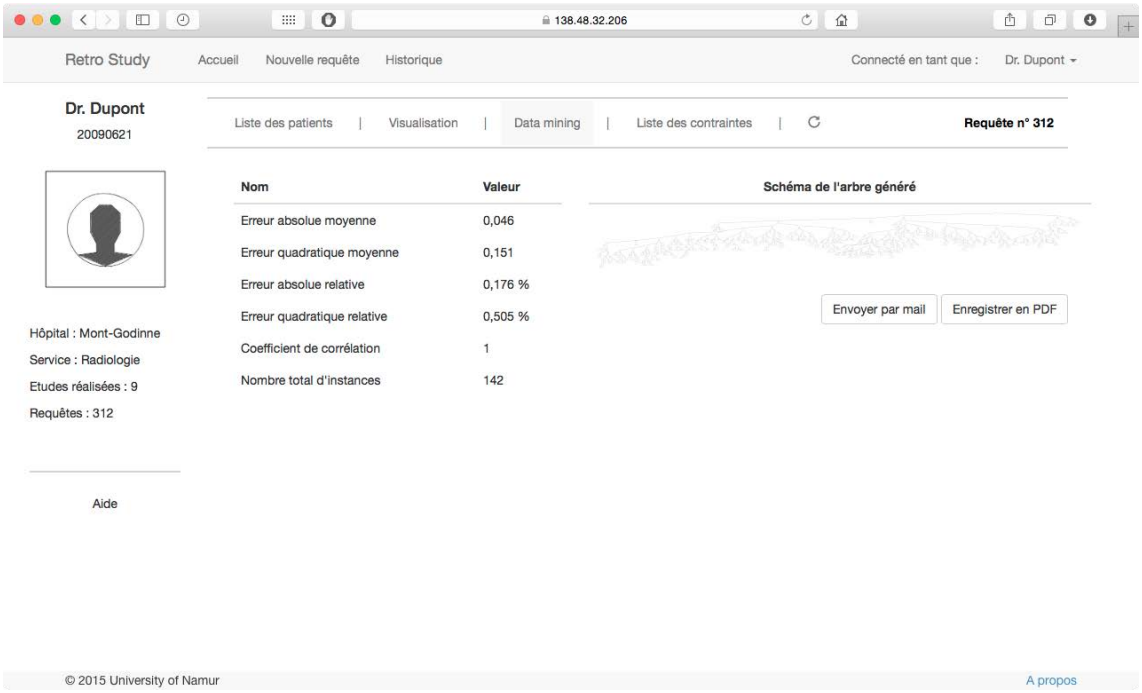

Figure 7.6 – Écran de statistique du data mining sur les données

7.4.4 Listes des Contraintes

Pour ne pas oublier quelles sont les contraintes qui ont été appliquées sur les données des patients, un écran permet de les afficher. Une illustration de cet écran est disponible à la figure 7.7.

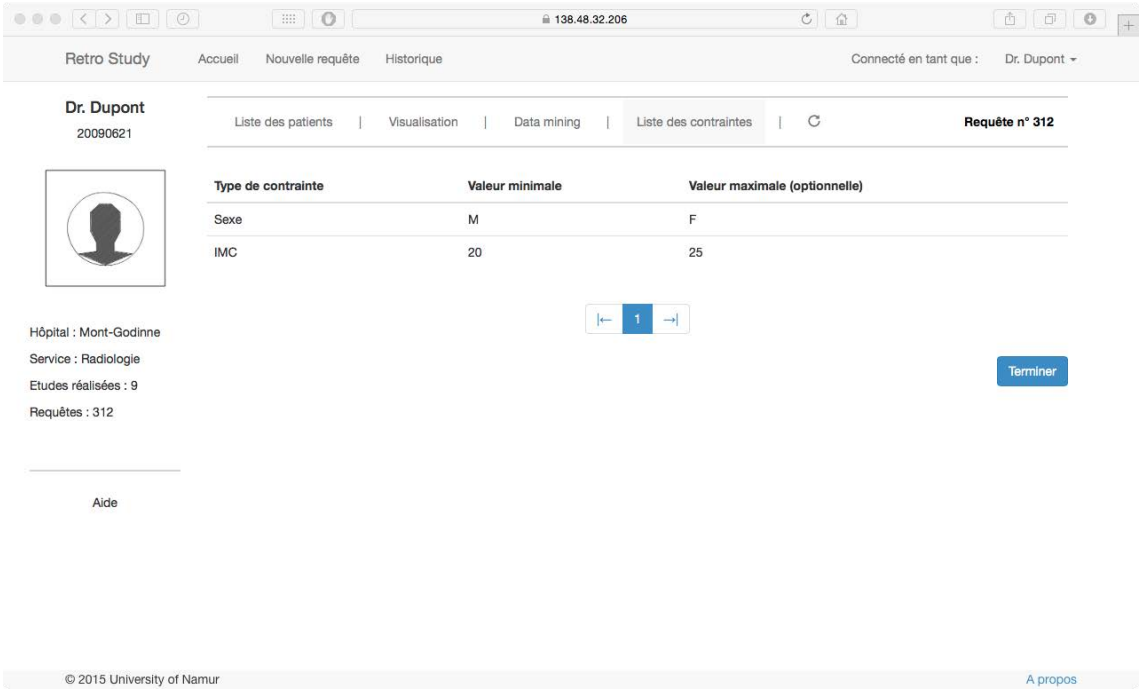

Figure 7.7 – Écran de rappel des contraintes appliquées lors de la requête

7.5 Déconnexion

En se déconnectant, l'utilisateur n'a plus accès au site web mais les traitements effectués seront toujours disponibles lors de sa prochaine connexion. Cela est possible par un enregistrement des opérations effectuées.

Chapitre 8

Conclusion

Ce chapitre donne une vue globale du mémoire et sa place dans le milieu médical et informatique. Dans un premier temps, une section fera le bilan des concepts utilisés tout au long du document ainsi que les points importants à retenir. Dans un deuxième temps, une autre section expliquera les travaux et les recherches qui peuvent s'inscrire dans la continuité de ce mémoire ainsi que les nombreuses améliorations du logiciel RetroStudy.

8.1 Bilan Global

Cette section tente de résumer les informations contenues dans ce mémoire au travers des deux premières sous-sections où le résumé sera divisé entre le niveau théorique et le niveau pratique. Une autre sous-section fera le point sur l'intégration des éléments présentés dans ce document.

8.1.1 Niveau Théorique

La partie sur l'état de l'art comporte trois chapitres qui traitent successivement des bases de données dans les hôpitaux, des ontologies et enfin de l'exploration des données. Toute cette partie, qui réalise une synthèse des éléments au niveau théorique, est présentée dans cette sous-section.

Au niveau des bases de données, les hôpitaux possèdent chacun leurs propres structures. Celles-ci pouvant faire intervenir une multitude de bases de données. La façon dont sont encodées les données est un problème lorsque l'institution hospitalière doit communiquer avec d'autres. Les professionnels de la santé l'ont compris et essaient tant bien que mal de se tourner vers des systèmes standardisés pour le stockage et la communication. Un enjeu majeur était de trouver le meilleur standard pour le stockage des données. Celui qui a été choisi pour le projet IFLEC est vMR. Il ne permet que le stockage des données mais propose un standard. Si un

seul hôpital l'utilise et que les autres gardent leur propre système, le problème de la communication sera toujours présent. Le défi suprême serait que tous les hôpitaux se basent sur le même standard.

Pour permettre aux médecins de stocker la connaissance des nombreux domaines de la santé (pathologies, médicaments, etc.), les ontologies viennent à leur secours. Conçues dans un premier temps pour structurer la connaissance d'un domaine, les ontologies apportent bien plus lorsqu'elles sont couplées avec un système d'aide à la décision. Elles structurent toute la connaissance d'un domaine ou d'une discipline en créant les liaisons entre les éléments. Ensuite, elles peuvent être considérées comme un standard au niveau de la sémantique en définissant les termes par exemple. Mais surtout, les ontologies, lorsqu'elles sont utilisées pour des requêtes de médecins, pourraient apporter des résultats supplémentaires. L'exemple des bactéries résistantes à un antibiotique (voir chapitre 2) donne plus de résultats avec les ontologies car ces dernières fournissent les liens, des équivalences entre certains éléments comme les antibiotiques par exemple.

Dans la continuité des ontologies, l'exploration des données (*Data Mining*) est utile pour découvrir l'information qui est cachée dans les gigantesques quantités de données. Plusieurs techniques peuvent être utilisées pour découvrir les liens cachés qui existeraient entre les données - au contraire des ontologies où les liens sont connus car définis dans l'ontologie - ou mettre au jour des regroupements de données. Des éléments statistiques peuvent aussi être générés par les algorithmes de data mining. Au niveau du domaine médical, le data mining pourrait prédire le risque de problème cardiaque en fonction de la santé d'un individu (voir chapitre 3).

8.1.2 Niveau Pratique

La seconde partie du mémoire qui traite de l'implémentation d'une application à un niveau plus pratique était également décomposée en plusieurs chapitres. Ces chapitres présentaient dans l'ordre l'infrastructure définie pour RetroStudy, la création sur mesure d'une base de données standardisée, les méthodes de data mining utilisées et enfin, les interfaces graphiques.

La première étape de la conception d'un outil informatique interactif d'analyse de données patients était de concevoir l'architecture. L'idée était d'intégrer les concepts de bases de données standardisées, d'ontologies et de data mining vus dans la première partie du document. Pour cela, il fallait créer une architecture qui intégrait ces trois concepts d'une manière très structurée de telle sorte que changer la base de données n'impose que des changements mineurs dans l'application web RetroStudy. Finalement, un schéma théorique de l'architecture globale visée a été élaboré. Au niveau de l'implémentation en elle-même, seule la partie sur les ontologies n'a pas pu être connectée au reste des composants. Malgré ce fait, tous les éléments ont été pensés pour accueillir très facilement et très prochainement les ontologies.

Afin de pouvoir traiter les données dans RetroStudy, un moyen de stockage était nécessaire. Il a été choisi d'utiliser le standard de vMR pour stocker les données. Comme le standard était conçu théoriquement et qu'aucune implémentation n'était déjà créée, il était indispensable de créer notre propre instance suivant le standard vMR. Une méthodologie a été ensuite imaginée pour permettre de passer du papier à la base de données. Le travail de remplissage des données était confiée à l'entreprise MIMS. Malheureusement, pour des raisons de timing trop serré, il n'a pas été possible de remplir cette base de données suffisamment tôt et l'utiliser comme il l'était initialement prévu. Pour compenser, une petite base de données a été conçue avec des données générées aléatoirement, suivant une distribution uniforme.

Le chapitre 6 présente la manière dont les algorithmes de data mining sont intégrés dans l'application web. Il a été décidé, avec l'aide précieuse d'un expert, d'implémenter pour commencer l'algorithme sur les arbres aléatoires, qui constituent l'une des méthodes de data mining appelée classification. Ce premier algorithme pourra donner une indication aux médecins utilisateurs de RetroStudy grâce à une représentation graphique des données ainsi que des données statistiques. D'autres méthodes et algorithmes devraient être intégrés dans le futur.

Dernier chapitre de la partie « Implémentation », les interfaces graphiques présentent d'une manière illustrative l'application web qui a été créée. L'idée de ce chapitre était, pour le lecteur, d'être en immersion dans le logiciel. La structuration voulait parcourir les fonctionnalités dans un ordre rationnel et présenter ainsi l'activité typique sur RetroStudy.

8.1.3 Intégration des Éléments

Pour rappel, l'architecture de l'application web devait intégrer tous les éléments présentés dans ce mémoire à savoir vMR, les ontologies et les méthodes de data mining. Comme il l'a déjà été mentionné, les ontologies et le système D2RQ qui permet la liaison avec la base de données relationnelle n'ont pas été mis en place. Les ontologies existent sur le web et ont été créées par des institutions médicales réputées. Le problème est la liaison de ces ontologies avec la base de données qui demande de fournir un travail considérable pour créer un fichier de mapping (voir D2RQ au chapitre 2). De ce fait, l'architecture finale du projet IFLEC, à l'heure actuelle, ressemble plutôt à celle illustrée à la figure 8.1.

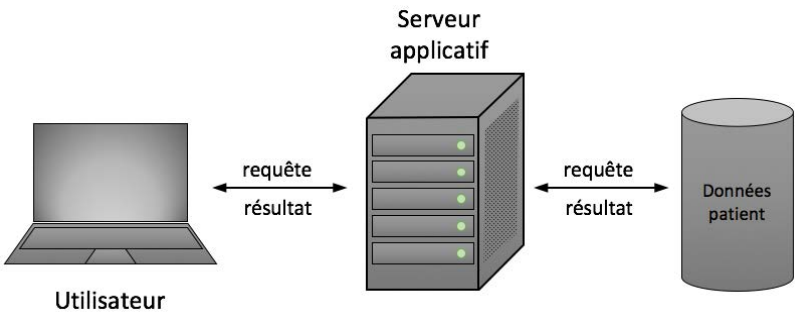

Figure 8.1 – Architecture réelle de RetroStudy

Une fois que les données seront correctement insérées dans la base de données qui a été créée, le travail de couplage des données avec les ontologies pourra commencer. En effet, le fichier de mapping est nécessaire pour pouvoir réaliser des requêtes basées non seulement sur les données mais aussi sur les ontologies.

Dans ces conditions, l'application web RetroStudy ne réalise pas l'ensemble des tâches qui étaient au départ prévues. Elle offre tout de même un bon aperçu de ce à quoi pourrait ressembler un outil interactif d'analyse de données patients avec une possibilité d'effectuer une exploration de données et retrouver de l'information cachée. Le domaine médical étant exigeant, rien ne doit être laissé au hasard. Il a été préféré de donner un aperçu plutôt que d'intégrer des composants qui ne seraient au final pas exploités.

8.2 Travaux Futurs

L'intégration des ontologies et des méthodes de data mining dans le domaine est quelque chose qui n'a à ce jour jamais été effectué. Ce mémoire a réalisé une synthèse des éléments et défini la marche à suivre pour élaborer un produit fini. En ce sens, RetroStudy a ouvert la voie vers un nouveau type de logiciel destiné à aider le patient dans ses tâches.

Le point faible de l'application est l'évaluation des résultats qui est inexistante. Évaluer les informations qui ressortent de la phase de data mining n'est possible qu'avec la coopération de médecins. Énoncer un niveau de fiabilité pour l'application est impossible. Malgré tout, les algorithmes de data mining fournissent à chaque exécution un taux d'erreur qui donnent une indication. Le processus de validation est une phase qui est nécessaire mais qui prendra du temps.

Pour clairement voir la puissance de l'outil, il est indispensable de travailler avec des données réelles pour analyser le comportement des algorithmes. Ainsi, en choisissant la base de données vMR (pour son côté standardisé) et les ontologies, le système informatique serait complet. D'autres améliorations sont à envisager :

Améliorations ponctuelles

- Ajouter un système de dictionnaire aux champs des formulaires ;
- Améliorer l'implémentation de la création dynamique de formulaires ;
- Gérer la recherche dans l'historique des requêtes ;
- Analyser et évaluer l'impact de RetroStudy ;
- Intégrer la base de données vMR ;
- Coupler de nombreuses ontologies à l'application pour augmenter le niveau de puissance ;
- Mettre en œuvre le système D2RQ ;
- Implémenter d'autres méthodes de data mining ;
- Etc.

Améliorations globales

— Lier plusieurs bases de données entre elles (par exemple celles de différents hôpitaux) ;
— Étendre le projet IFLEC à tous les hôpitaux belges et travailler ainsi sur d'immenses collections de données ;
— Coupler RetroStudy aux logiciels existants dans les hôpitaux ;
— Permettre la création d'une étude clinique à partir des résultats de RetroStudy ;
— Etc.

L'application RetroStudy reste avant tout un prototype. Son potentiel théorique n'est pas à démontrer. Il suffit de regarder le gain entre la même requête effectuée avec ou sans les ontologies. Plus encore, étendre le système à tous les hôpitaux du pays pourrait changer la vision de la médecine. Les institutions partageraient leurs informations et l'exploration de données ne se ferait que plus facilement.

La matière étudiée dans ce mémoire est en constante évolution et ce travail sur la recherche informatique au niveau médical permettrait à l'avenir de comprendre mieux les liens entre certains éléments. Ce mémoire apporte, avec conviction, un éclairage sur les concepts d'ontologies et de data mining pour permettre à la recherche de continuer dans la mise en œuvre d'un tel outil.

Bibliographie

[1] Site internet du Centre Hospitalier Godinne – Dinant. http://www.uclmontgodinne.be, 12 février 2015.

[2] Site internet de la société MIMS. http://www.mims.be, 12 février 2015.

[3] P. Benats. Administrative management of clinical studies in hospitals. Master's thesis, Université de Namur, Belgique, 2014.

[4] Projet IFLEC, PReCISE, and MIMS. Rapport scientifique et technique (09/2013 - 09/2014). Université de Namur, Belgique, 2014.

[5] Agence Wallonne des Télécommunications. Dossier médical informatisé (DMI) : évolution et conditions à remplir. http://www.awt.be/web/dem/index.aspx?page=dem,fr,050,020,001, 2003.

[6] Health Level Seven - Informations générales. http://www.hl7.org, Decembre 2014.

[7] C. de Quirini. Analyse des standards HL7 et KMEHR et développement d'un outil de conversion. Master's thesis, Université de Namur, Belgique, 2014.

[8] Corepoint Health. http://www.corepointhealth.com/sites/default/files/whitepapers/hl7-v2-v3-evolution.pdf, Juin 2015.

[9] HL7 Reference Information Model. http://www.hl7.org/implement/standards/rim.cfm, 2015.

[10] Clinical Document Architecture. http://searchhealthit.techtarget.com/definition/Clinical-Document-Architecture-CDA, 2015.

[11] eHealth. Implémentation d'un standard belge (KMEHR). https://www.ehealth.fgov.be/standards/kmehr, 2015.

[12] SMALS. Échange standardisé en ligne de données de santé. https://www.smals.be/fr/realisations/projets/kmehr, 2013.

[13] T. Borloo and N. Pien. A comparison between HL7 & KMEHR. Slides, 2014.

[14] Organisation HL7. Informations techniques sur le standard VMR et liens vers les fichiers à télécharger. http://wiki.hl7.org/index.php?title=HL7_CDS_Standards, Septembre 2014.

[15] V-S. Gomoi. *Interoperability of heterogeneous sources of patient clinical data and multiple clinical practice guidelines formalisms to leverage computerized decision support*. PhD thesis, Faculty of Automation and Computers, Politehnica University of Timoşoara, Romania, 2012.

[16] K. Kawamoto, D. Shields, A. K. McIntyre, Y. Bao, H. R. Strasberg, P. R. Tattam, S. Bolte, P. Scott, K. Boone, Z. Liu, C. Melo, N. Hulse, J. Basilakis, R. Worden, D. Chertcoff, C. Curtis, G. Del Fiol, E. Fry, J-C. Dufour, and

L. Charlois. *Virtual Medical Record (vMR) for Clinical Decision Support - Domain Analysis Model*. PhD thesis, University of Utah, Salt Lake City, USA, 2011.

[17] Description de openEHR. http://www.openehr.org/what_is_openehr, 2015.

[18] A. Venot, A. Burgun, and C. Quantin. *Informatique Médicale, e-Santé - Fondements et applications*. Springer, 2013.

[19] J-R. Scherrer and S. Spahni. Health Information System Architecture (HISA) and its Middleware Models. Technical report, Geneva Faculty of MEdecine, University of Geneva, Switzerland, 1999.

[20] BRIDG. Information sur le standard BRIDG. http://bridgmodel.nci.nih.gov, 2015.

[21] EDIFact. Information sur le standard EDIFact. http://www.edifact.fr, 2015.

[22] CDISC. *Clinical Data interchange Standards Consortium - Strength through Collaboration*. http://www.cdisc.org, 2015.

[23] Siemens. Soarian Clinicals - Designing the hospital of the future. http://www.healthcare.siemens.com/siemens_hwem-hwem_ssxa_websites-context-root/wcm/idc/groups/public/@global/@healthit/documents/download/mdaw/nde1/~edisp/soarian-clinicals-designing-the-hospital-of-the-future-00304737.pdf, 2015.

[24] Corilius. Careconnect, le DMI de l'avenir. http://www.careconnect.be, 2015.

[25] OpenCDS.org. Open Clinical Decision Support - Tools and Ressources. http://www.opencds.org, 2015.

[26] InteliChart. InteliChart - Connected Health. http://www.intelichart.com, 2015.

[27] *Dictionnaire Français Illustré*. Larousse, 2006.

[28] Reverso. Définitions de mots français. http://dictionnaire.reverso.net/francais-definition.

[29] Technolangue.net. Qu'est-ce qu'une ontologie ? http://www.technolangue.net/imprimer.php3?id_article=280, 2015.

[30] S. Zefouni. *Aide à la conception de workflows personnalisés : application à la prise en charge à domicile*. PhD thesis, Université Toulouse III Paul Sabatier, France, 2012.

[31] M. Horridge. *A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools*. The University of Manchester, 2009.

[32] W3C. Présentation du standard RDF. http://www.w3.org/RDF.

[33] P. Thiran. Web Technologies. Chapter 6 : Semantic Web Technologies - RDF and SPARQL, 2015.

[34] A. Bodart and K. Evrard. ArThUR : Un outil d'aide à la manipulation d'ontologies et de la logique de markov. Master's thesis, Université de Namur, Belgique, 2014.

[35] W3C. Présentation du standard owl. http://www.w3.org/2001/sw/wiki/OWL.

[36] Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language overview. http://www.w3.org/TR/owl-features/, 2004.

[37] D. Connolly, F. van Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. Stein. DAML+OIL Reference Description. http://www.w3.org/TR/daml+oil-reference, 2001.

[38] Cambridge Semantics. Introduction to the Semantic Web. http://www.cambridgesemantics.com/semantic-university/introduction-semantic-web, 2015.

[39] GitHub. The OntoUML Lightweight EDitor. https://github.com/nemo-ufes/ontouml-lightweight-editor, 2015.

[40] The Apache Software Foundation. Jena Ontology API. https://jena.apache.org/documentation/ontology, 2015.

[41] The Eclipse Foundation. Eclipse Project - Mars Release. https://eclipse.org, 2015.

[42] M. Magnin. Le Web Sémantique. http://www-igm.univ-mlv.fr/~dr/XPOSE2009/Le%20Web%203.0/concepts.html, 2010.

[43] Stanford Center for Biomedical Informatics Research. Protégé - Ontology editor and framework for building intelligent systems. http://protege.stanford.edu, 2015.

[44] The NeOn Technologies Foundation. NeOn Ontology tool. http://neon-toolkit.org/wiki/Main_Page.html, 2010.

[45] Projet D2RQ. Information sur la plateforme D2RQ. http://d2rq.org, Janvier 2015.

[46] A. Assélé Kama, G. Mels, R. Choquet, J. Charlet, and M-C Jaulent. Une approche ontologique pour l'exploitation de données cliniques. *IC 2010*, 2010.

[47] M. Cuggia, J-C. Dufour, O. Zekri, I. Gibaud, C. Garde, C. Bohec, R. Duvauferrier, D. Fieschi, P. Besana, L. Charlois, A. Bourde, N. Garcelon, J. Laurent, M. Fieschi, and O. Dameron. Système sémantiquement interopérable de sélection semi-automatique des patients éligibles aux essais thérapeutiques en cancérologie. *IRBM*, 2012.

[48] International Health Terminology Standards Development Organisation (IHTSDO). SNOMED - The global language of healthcare. http://www.ihtsdo.org/snomed-ct, 2015.

[49] C. Eccher, A. Ferro, and D. M. Pisanelli. An ontology of therapies. *eHealth 2009 - LNICST 27*, 2010.

[50] D. Cohen. How facebook manages a 300-Petabyte data warehouse, 600 Terabytes per day. http://www.adweek.com/socialtimes/orcfile/434041, 2014.

[51] J. Han, M. Kamber, and J. Pei. *Data Mining : Concepts and Techniques*. Kaufmann, 2012.

[52] X-S. Zhong, P. Schuette, S. Komo, and A. Parfionovas. An evaluation of data mining methods applied to adverse events for clinical trials. Poster, 2012.

[53] S. Shajahaan, S. Shanthi, and V. Chitra. Application of data mining techniques to model breast cancer datada mining techniques to model breast cancer dataata mining techniques to model breast cancer data. *International Journal of Emerging Technology and Advanced Engineering (IJETAE)*, 2013.

[54] M. Fieschi. Data Mining, fouille de données : Concepts et techniques. Cours de la faculté de médecine, 2006.

[55] A. Juditsky. Introduction au Data Mining. Cours pour Sciences de l'ingénieur, 2015.

[56] F. Voznika and L. Viana. Data Mining Classification. Applied Algorithms Course, 2007.

[57] B. Liaudet. Cours de Data Mining 3 : modélisation - présentation générale. Cours pour l'Option Ingénierie d'Affaires et de Projets - Finance, 2008.

[58] G. Calas. Etudes des principaux algorithmes de data mining. Cours de spécialisation Sciences cognitives et Informatique avancée, 2009.

[59] Projet Weka. Information sur la librairie Weka. `http://www.cs.waikato.ac.nz/ml/weka`, Décembre 2014.

[60] The R Foundation. The R Project for Statistical Computing. `https://www.r-project.org`, 2015.

[61] Y. Zhao. Introduction to data mining with R and data import/export in R. Slides, 2014.

[62] F. E. Bekri and A. Govardhan. Association of data mining and healthcare domain : Issues and current state of the art. *Global Journal of Computer Science and Technology*, 2011.

[63] S. Kharya. Using data mining techniques for diagnosis and prognosis of cancer disease. *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, 2012.

[64] X. Cao, K. B. Maloney, and V. Brusic. Data mining of cancer vaccine trials : a bird's-eye view. *Immunome Research*, 2008.

[65] Projet Java. Information sur la technologie Java. `http://www.oracle.com/fr/java/overview/index.html`, Janvier 2015.

[66] Projet Bootstrap. Informations générales sur le framework. `http://getbootstrap.com`, Novembre 2014.

[67] Projet Highcharts. Information sur la librairie Highcharts. `http://www.highcharts.com`, Décembre 2014.

[68] Projet Graphviz. Information sur la librairie Graphviz. `http://www.graphviz.org`, Décembre 2014.

[69] SPARX Systems. Informations sur le logiciel Enterprise Architect. `http://www.sparxsystems.com.au`, Septembre 2014.

[70] J-L. Hainaut. *Bases de Données - Concepts, utilisation et développement*. Dunod, 2012.

[71] J-L. Hainaut. Cours de bases de données - Le modèle Entité-Association étendu. Chapitre 15, Septembre 2014.

[72] J-L. Hainaut. Cours de bases de données - Conception logique Relationnelle. Chapitre 18, Septembre 2014.

[73] L. Audibert. Cours de UML avancé en ligne. `http://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-classes`, Septembre 2014.

[74] Oracle. Présentation du logiciel MySQLWorkBench. `https://www.mysql.fr/products/workbench`, Octobre 2014.

[75] G. Hunault. Découpage en classes et discrétisation. http://www.info.univ-angers.fr/~gh/wstat/discr.php.

[76] Technopedia. Définition du protocole HTTPS. http://www.techopedia.com/definition/5361/hypertext-transport-protocol-secure-https, Juin 2015.

[77] Département Paramédical Sainte-Elisabeth. *Normes de biologie clinique*. Hénallux, Namur, Belgique, 2014.

Annexe A

Logiciels Associés

Cette annexe présente les interfaces graphiques de plusieurs logiciels qui sont liés au chapitre 1. Ils se basent sur l'un ou l'autre standard présent dans le chapitre.

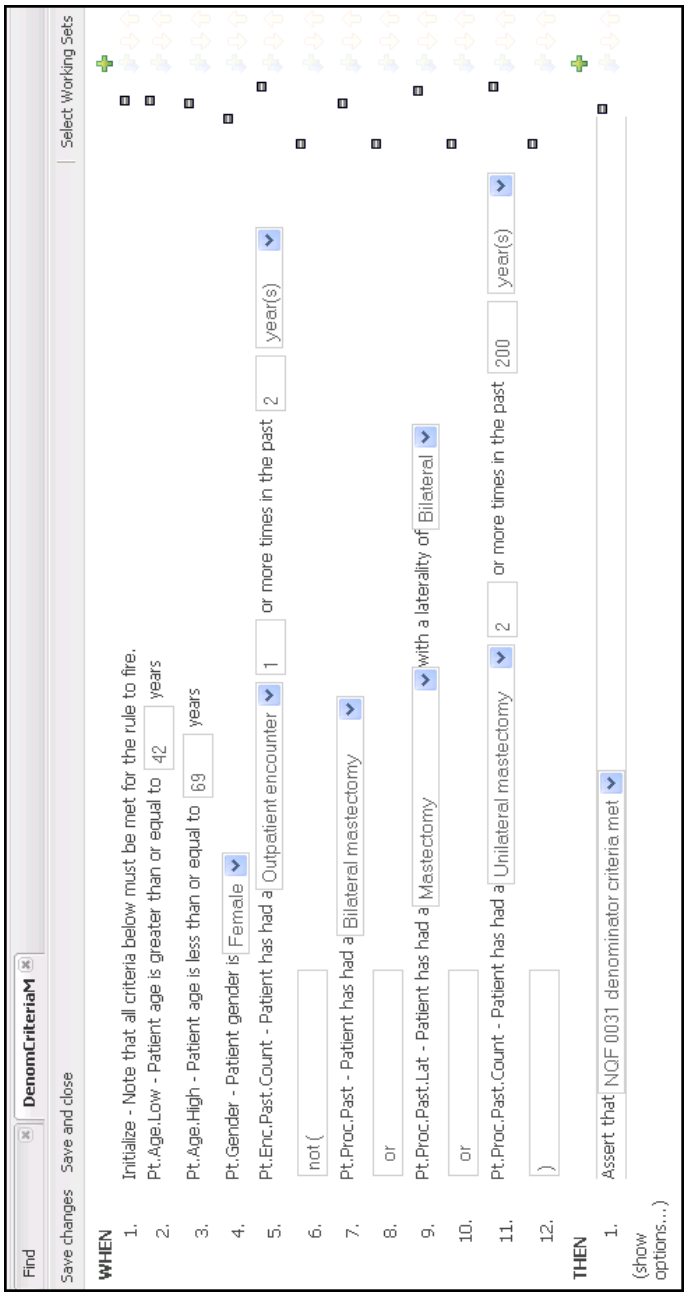

FIGURE A.1 – Illustration de OpenCDS

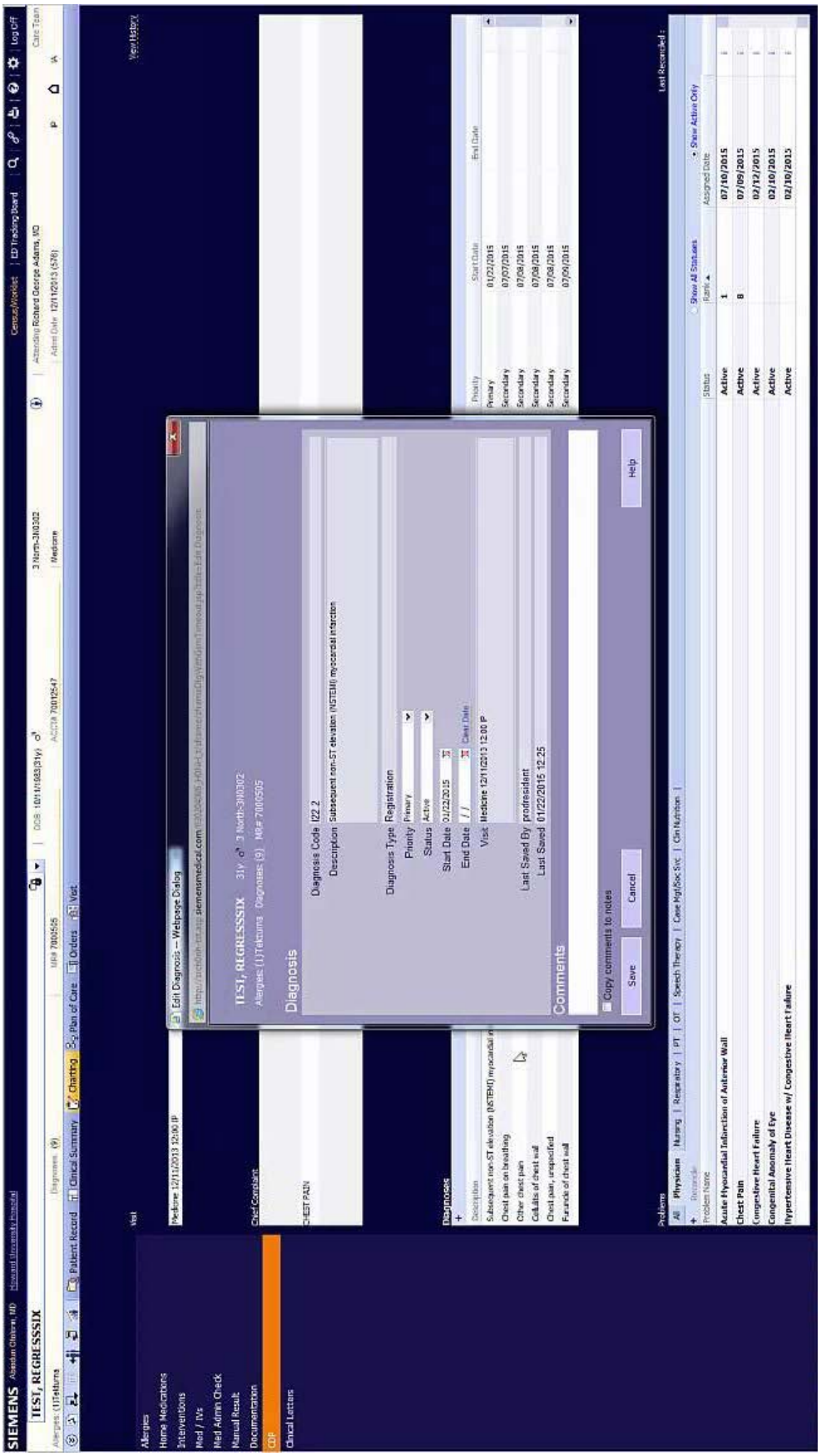

Figure A.2 – Illustration de Soarian (Siemens)

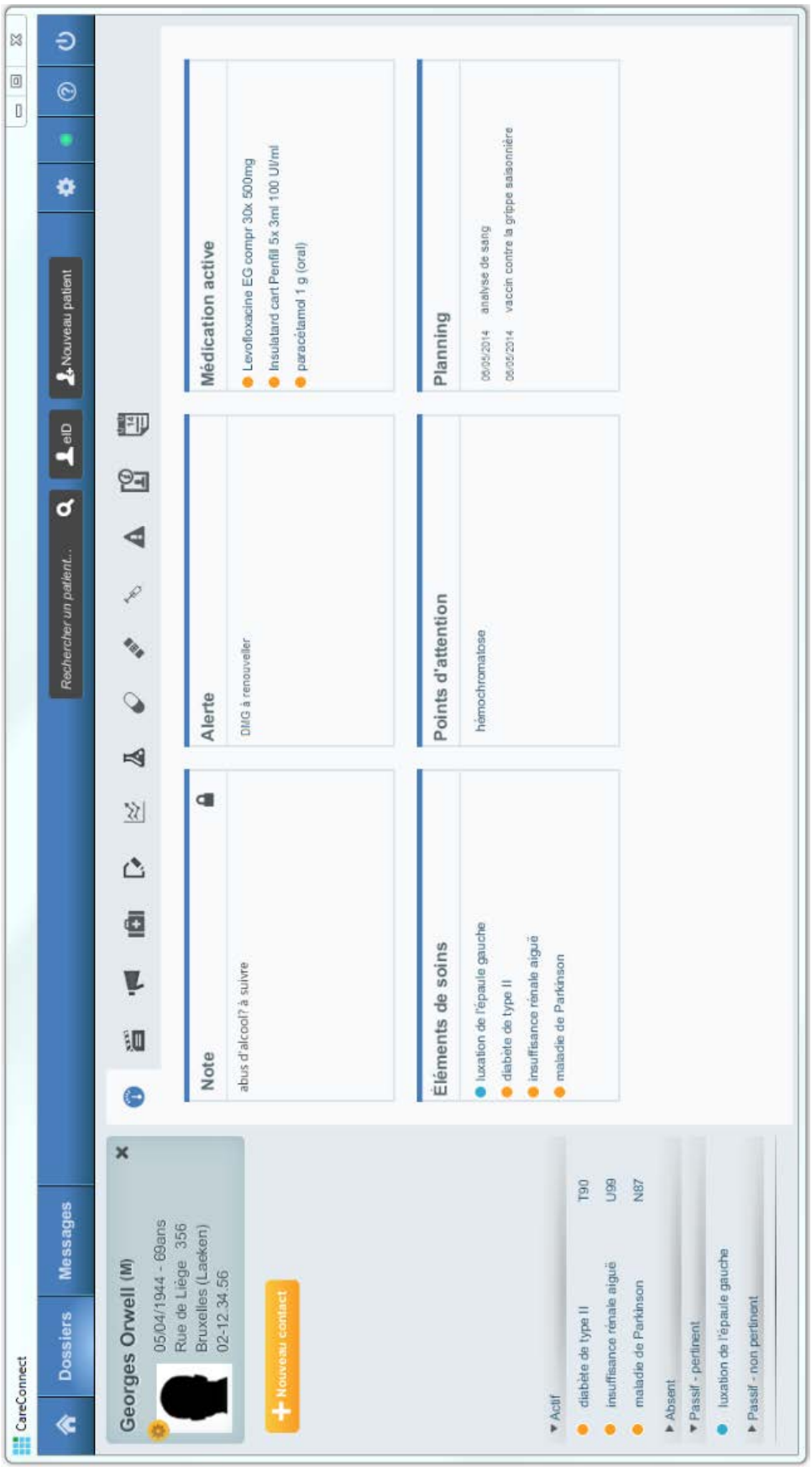

FIGURE A.3 – Illustration de CareConnect

Figure A.4 – Illustration de InteliChart

Annexe B

Diagrammes Initiaux de vMR

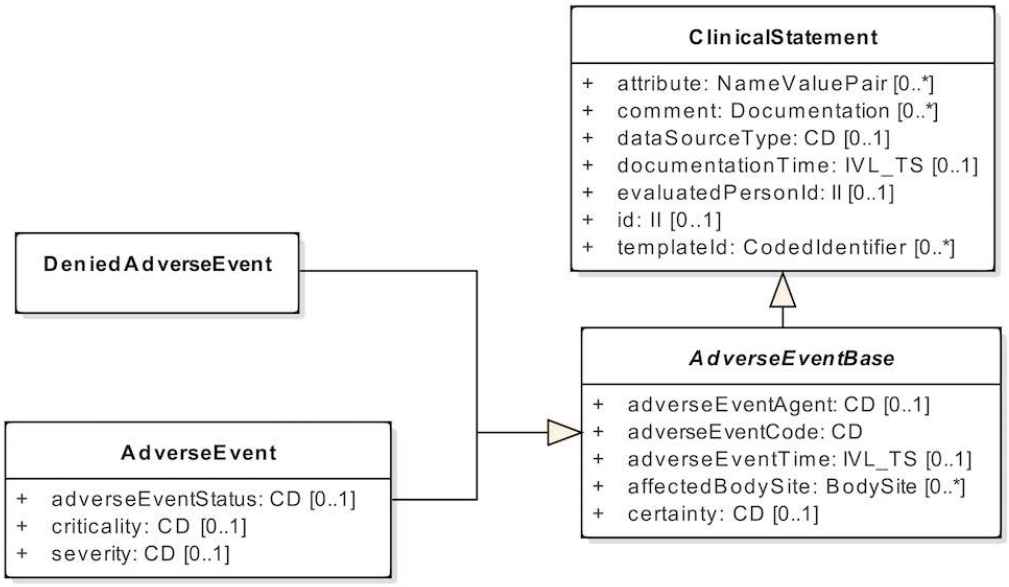

FIGURE B.1 – Représentation initiale du standard vMR (1/19)

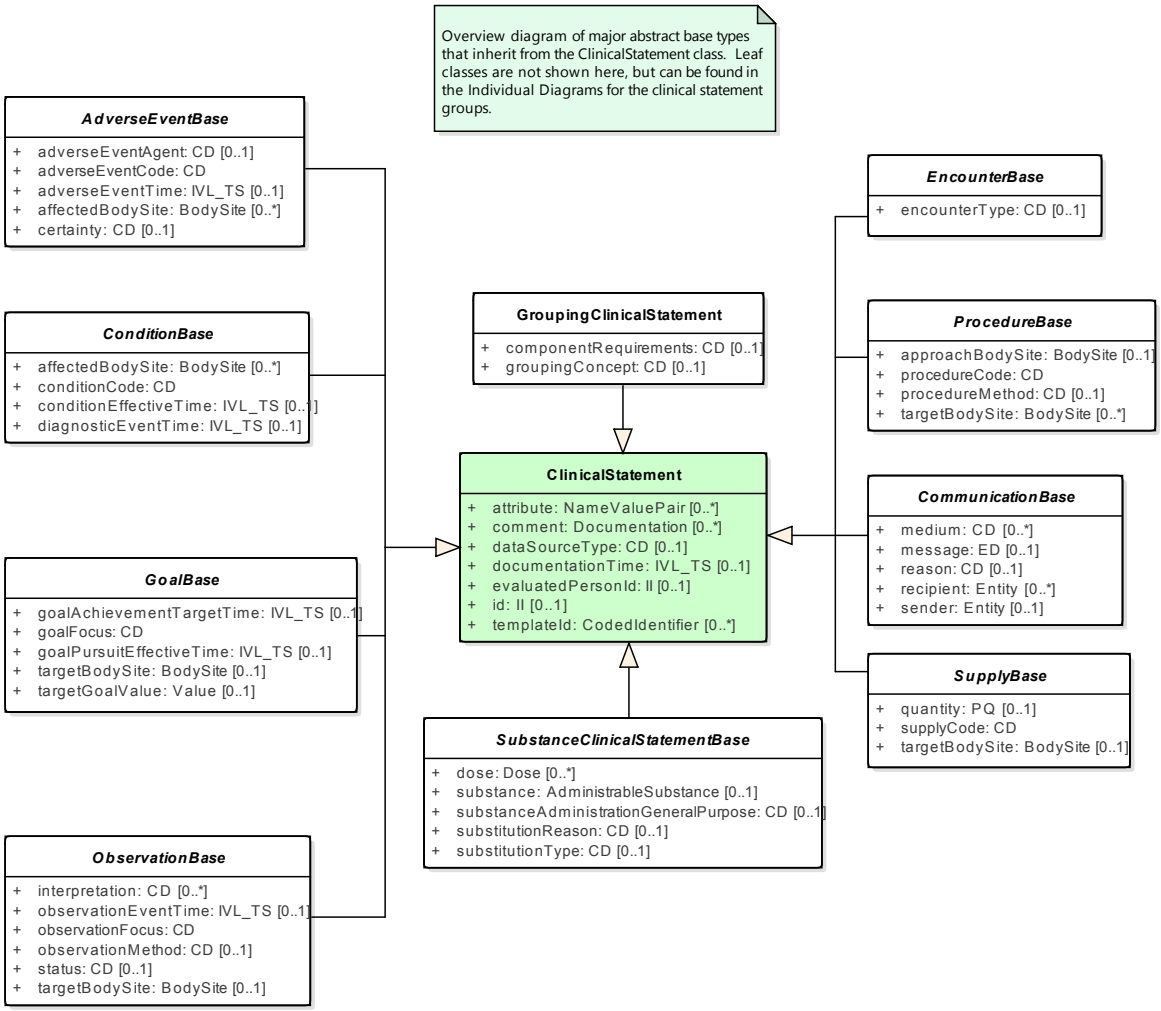

Figure B.2 – Représentation initiale du standard vMR (2/19)

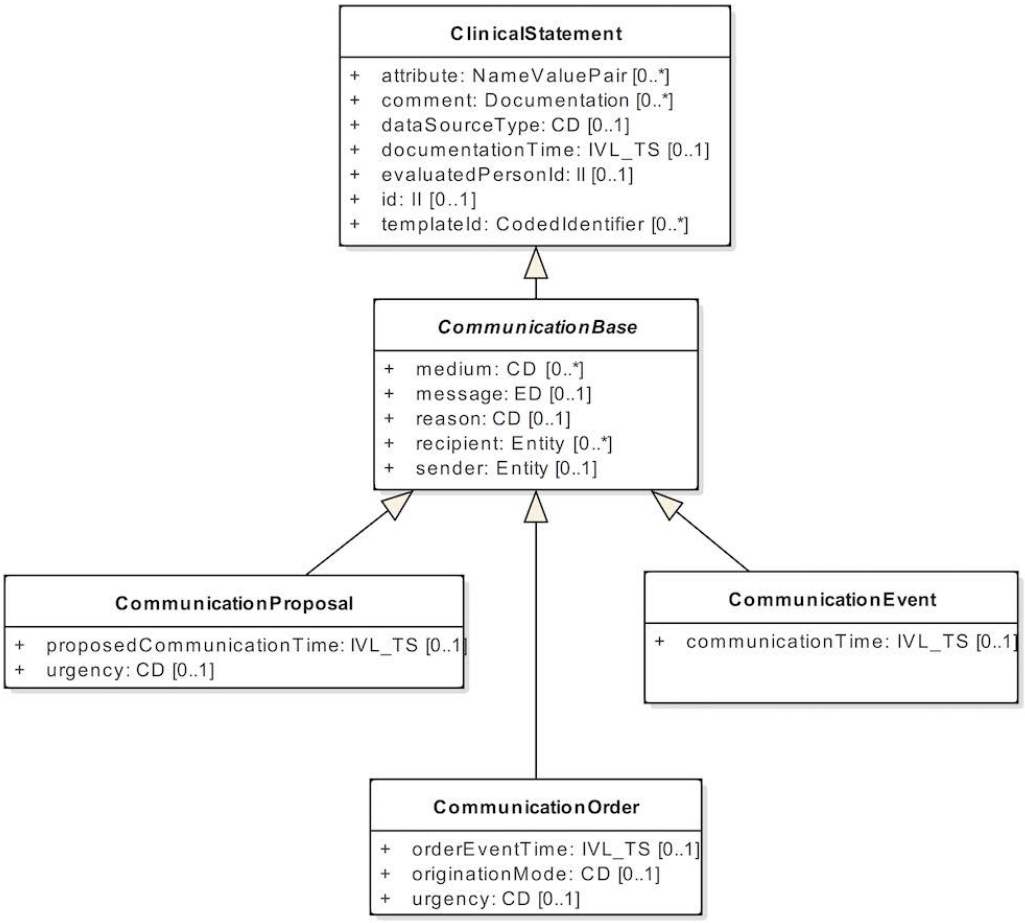

FIGURE B.3 – Représentation initiale du standard vMR (3/19)

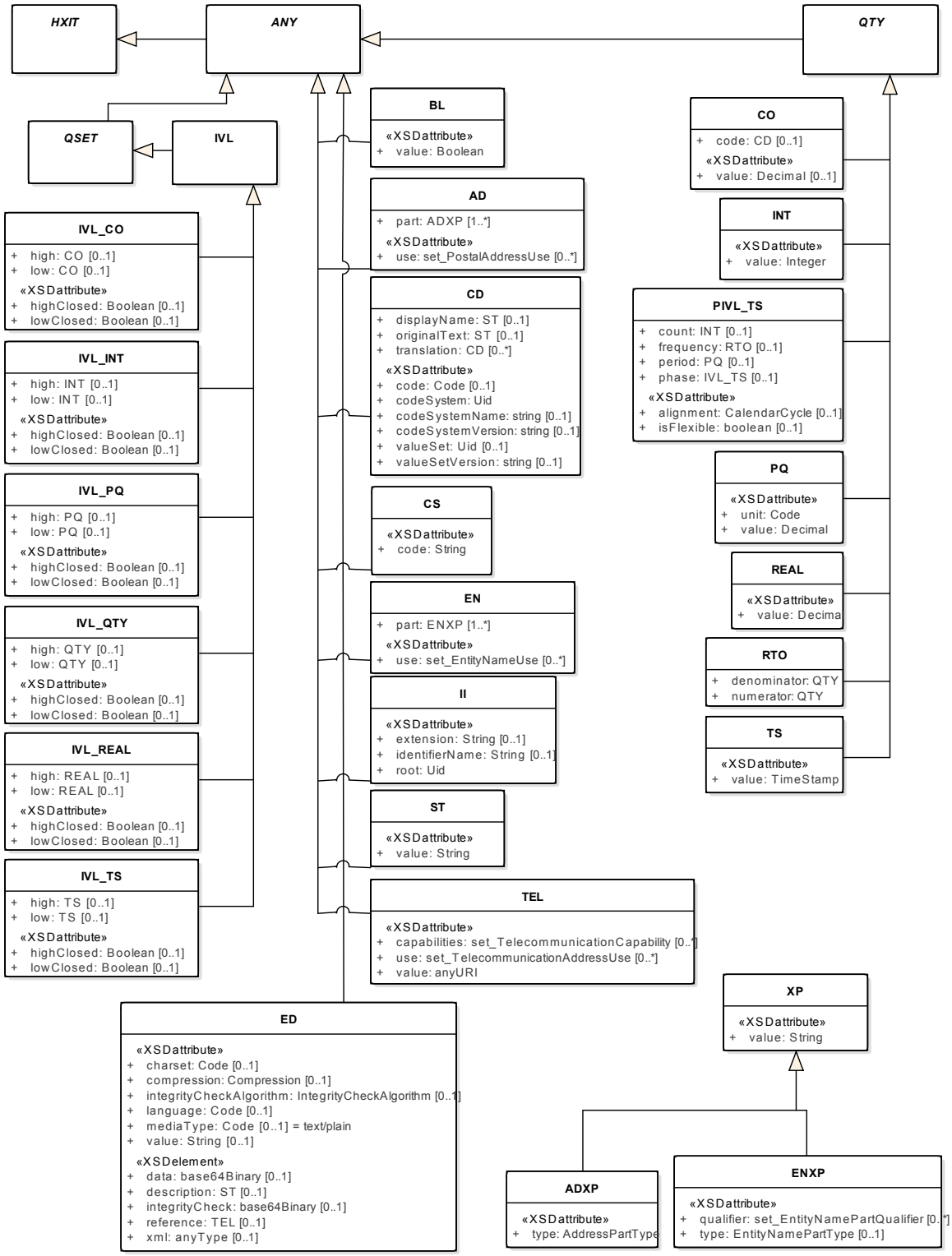

Figure B.4 – Représentation initiale du standard vMR (4/19)

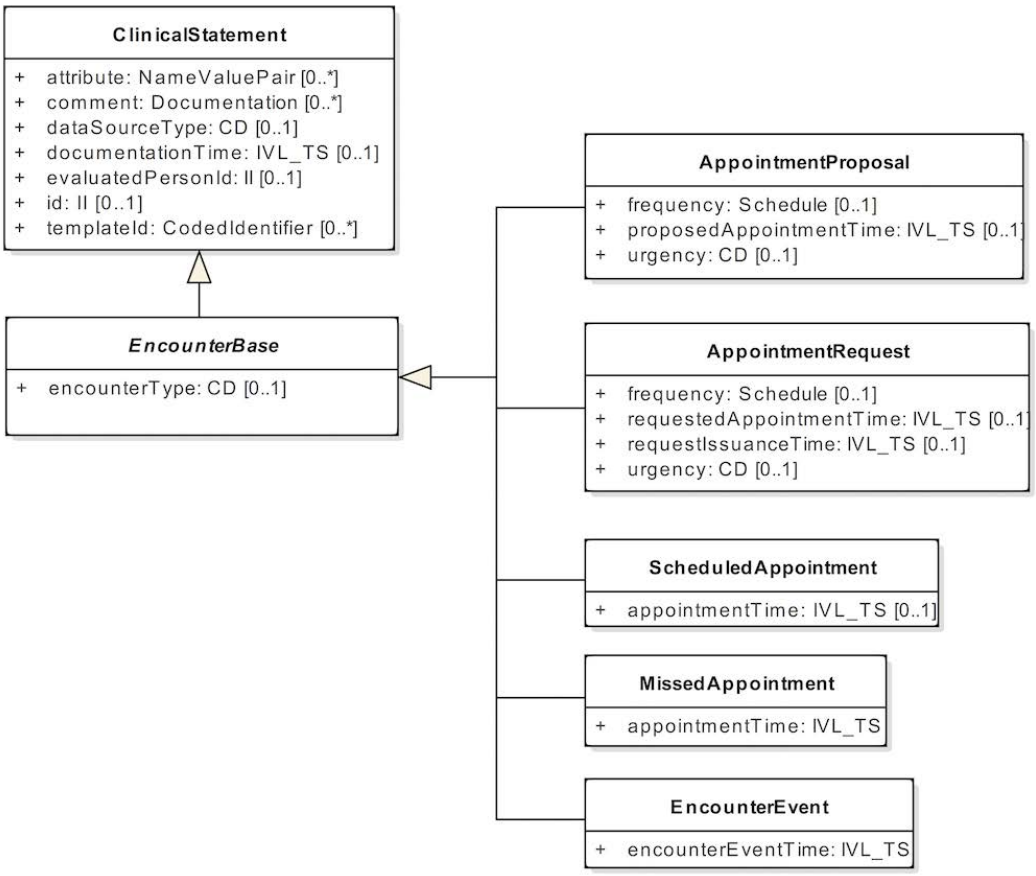

FIGURE B.5 – Représentation initiale du standard vMR (5/19)

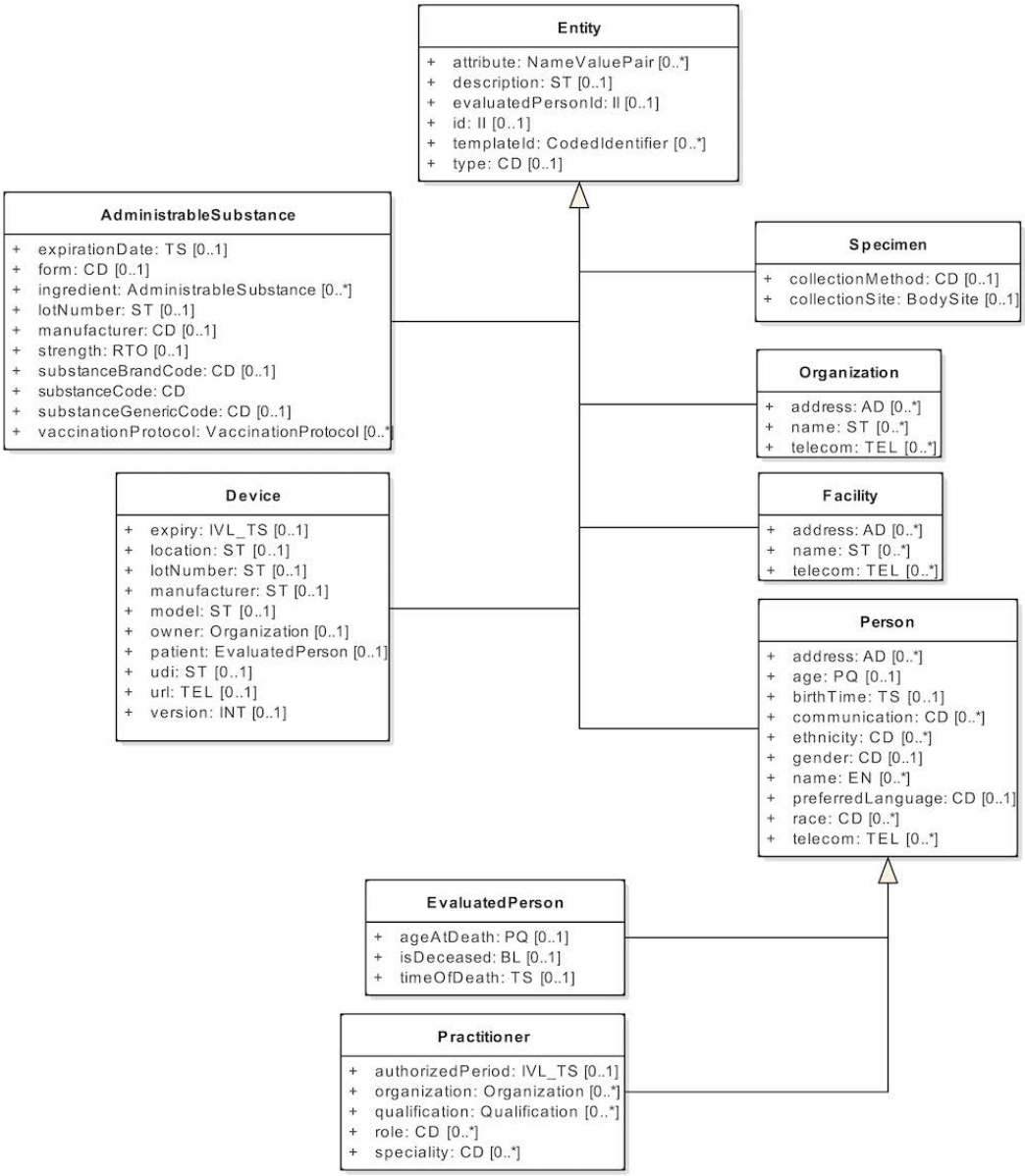

Figure B.6 – Représentation initiale du standard vMR (6/19)

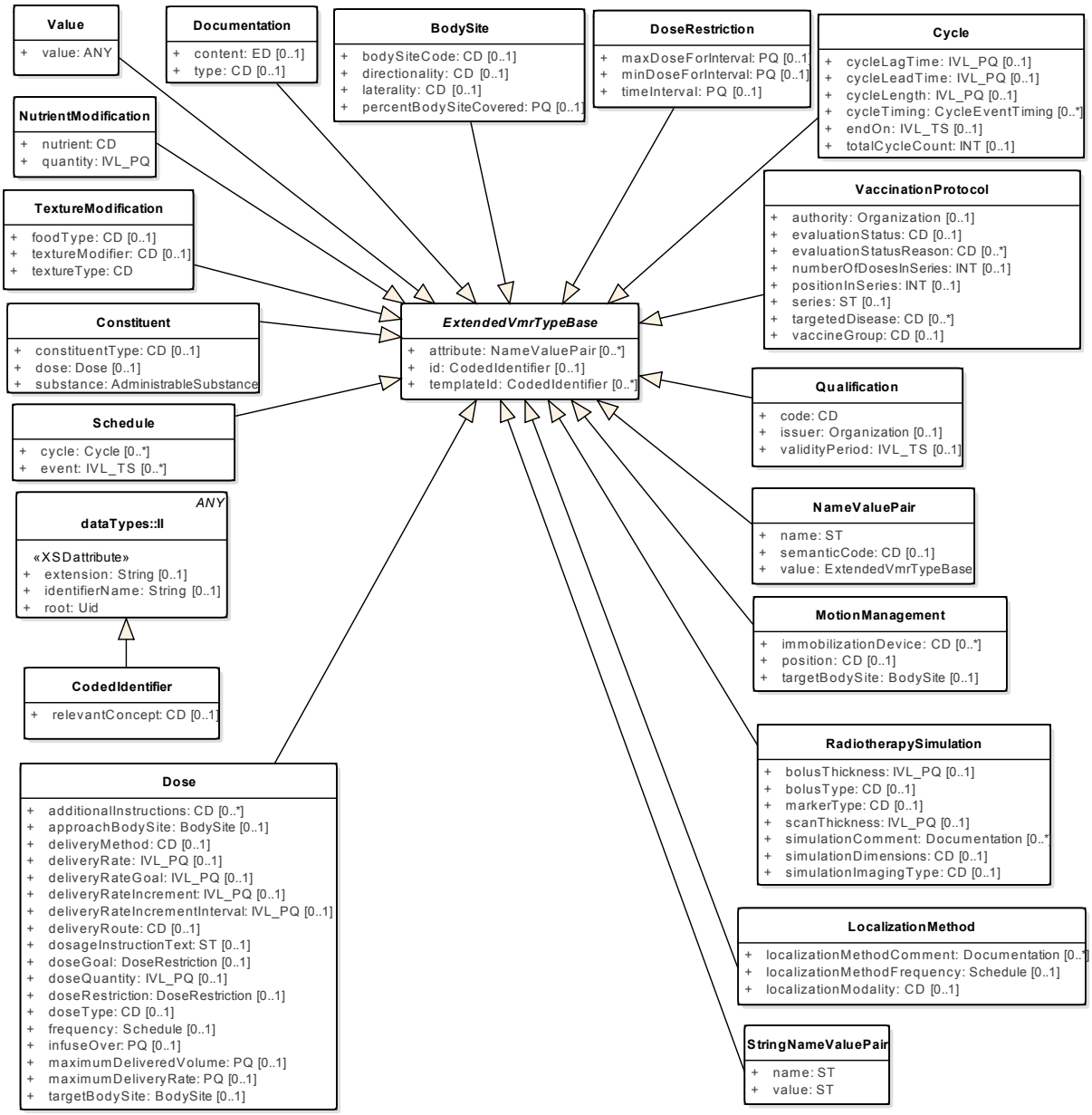

FIGURE B.7 – Représentation initiale du standard vMR (7/19)

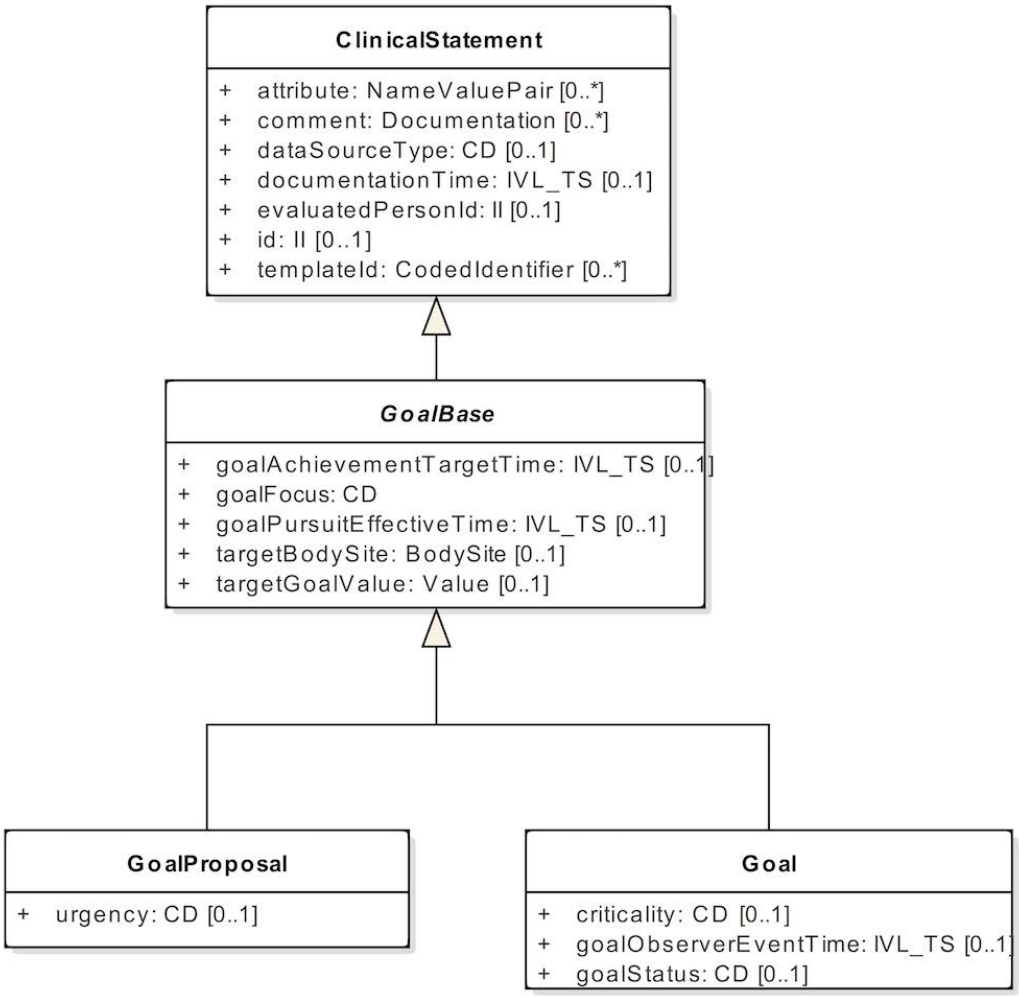

Figure B.8 – Représentation initiale du standard vMR (8/19)

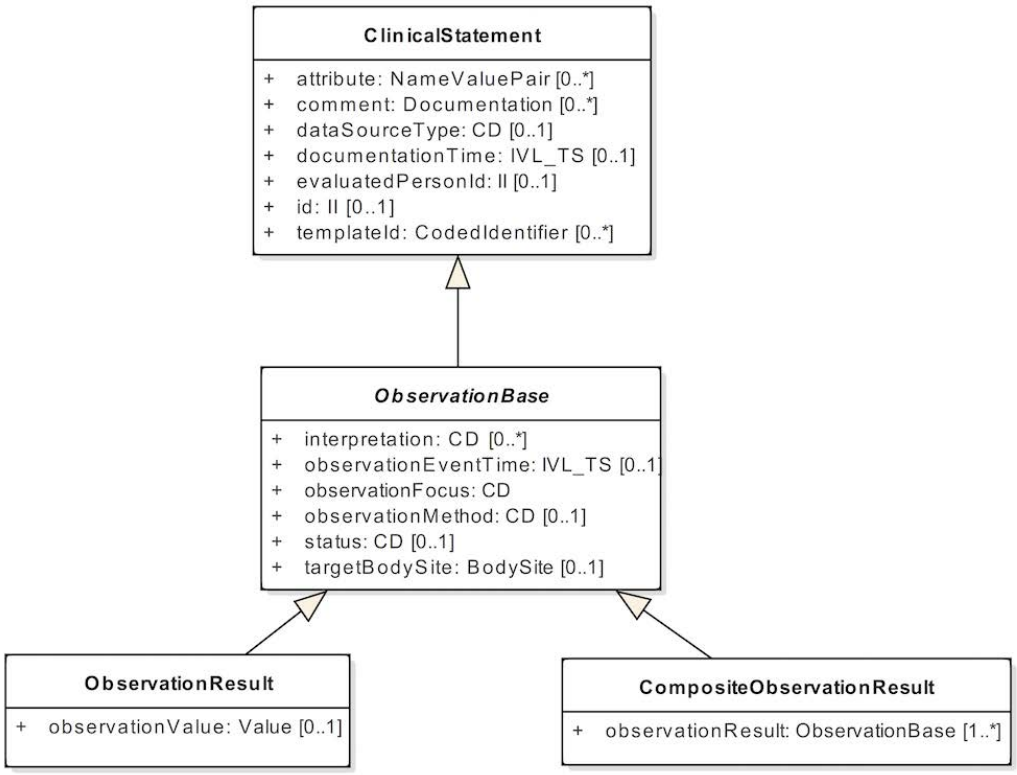

FIGURE B.9 – Représentation initiale du standard vMR (9/19)

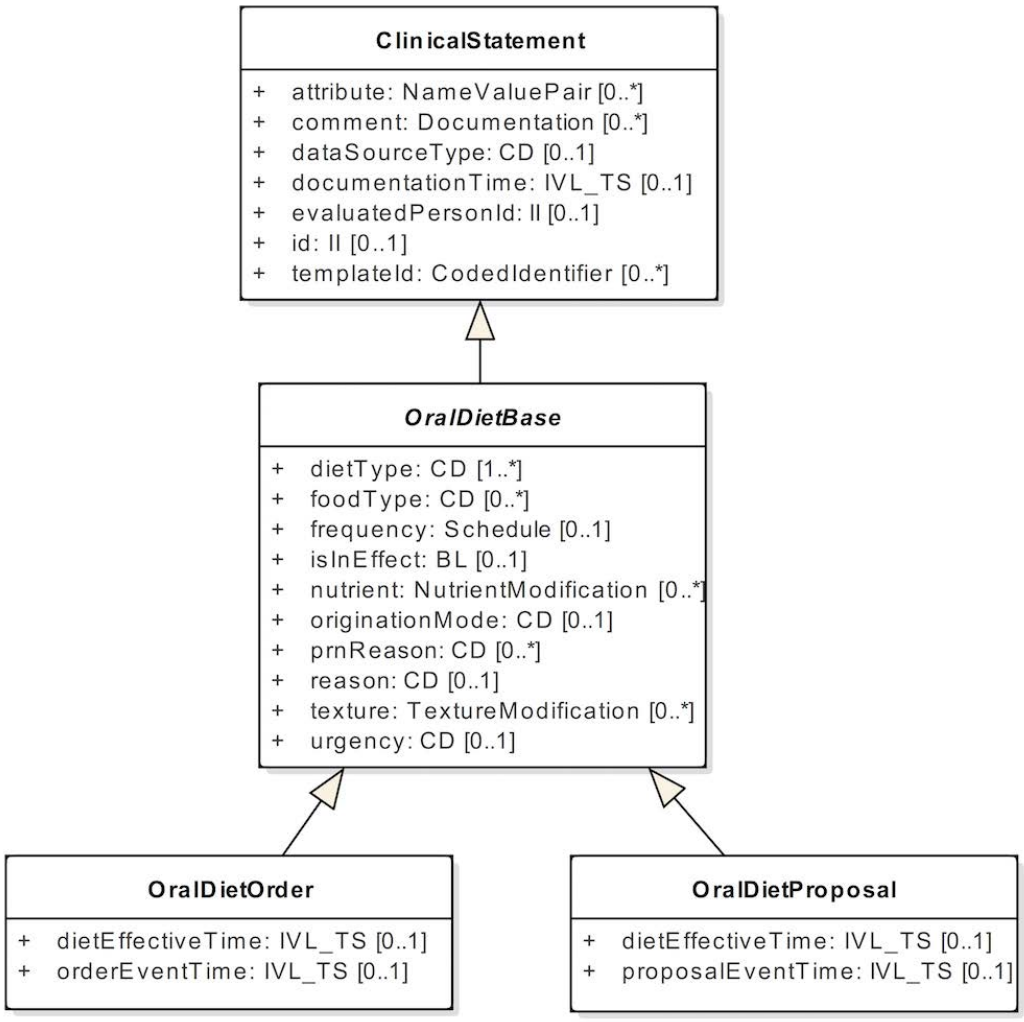

Figure B.10 – Représentation initiale du standard vMR (10/19)

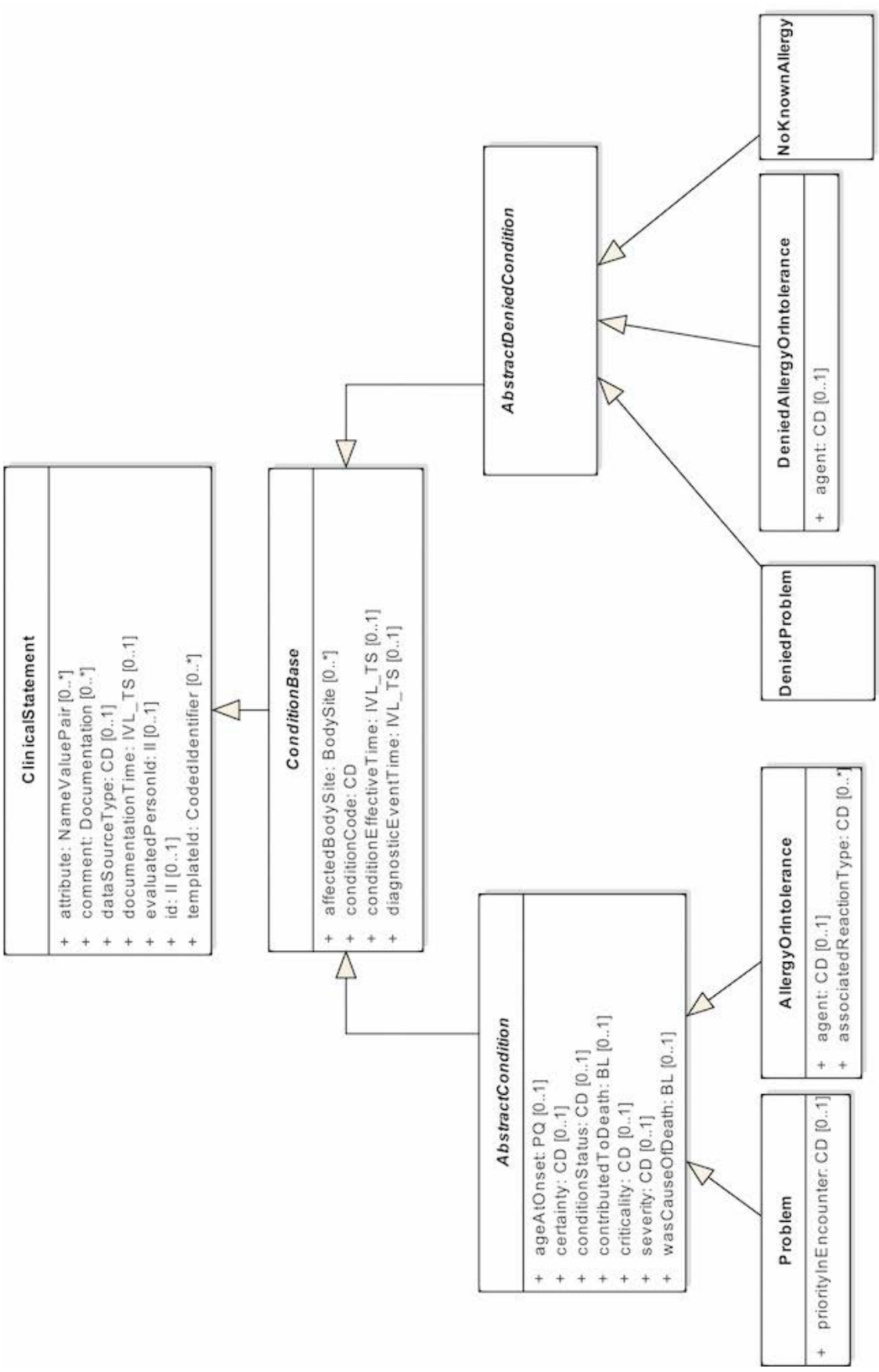

FIGURE B.11 – Représentation initiale du standard vMR (11/19)

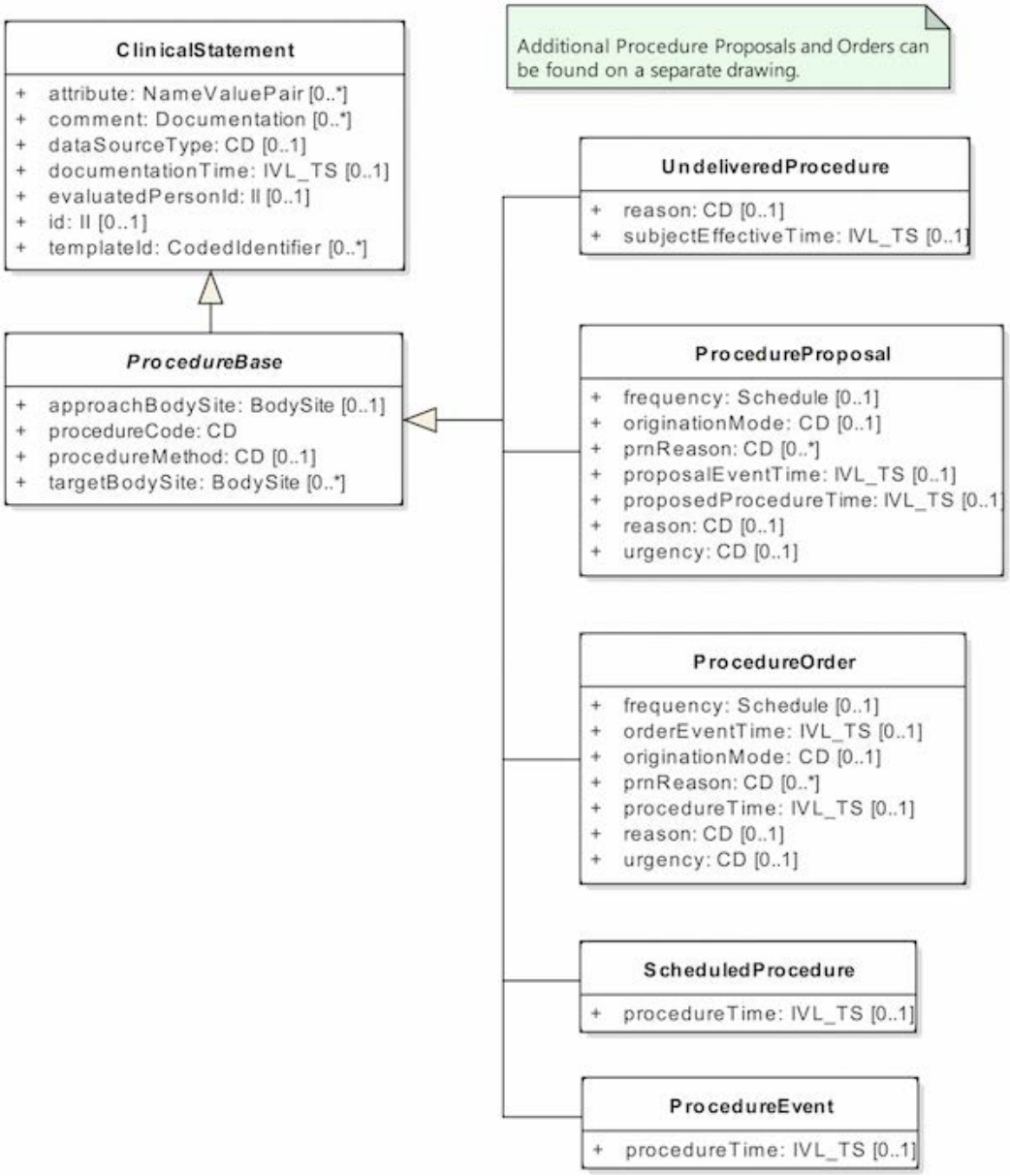

Figure B.12 – Représentation initiale du standard vMR (12/19)

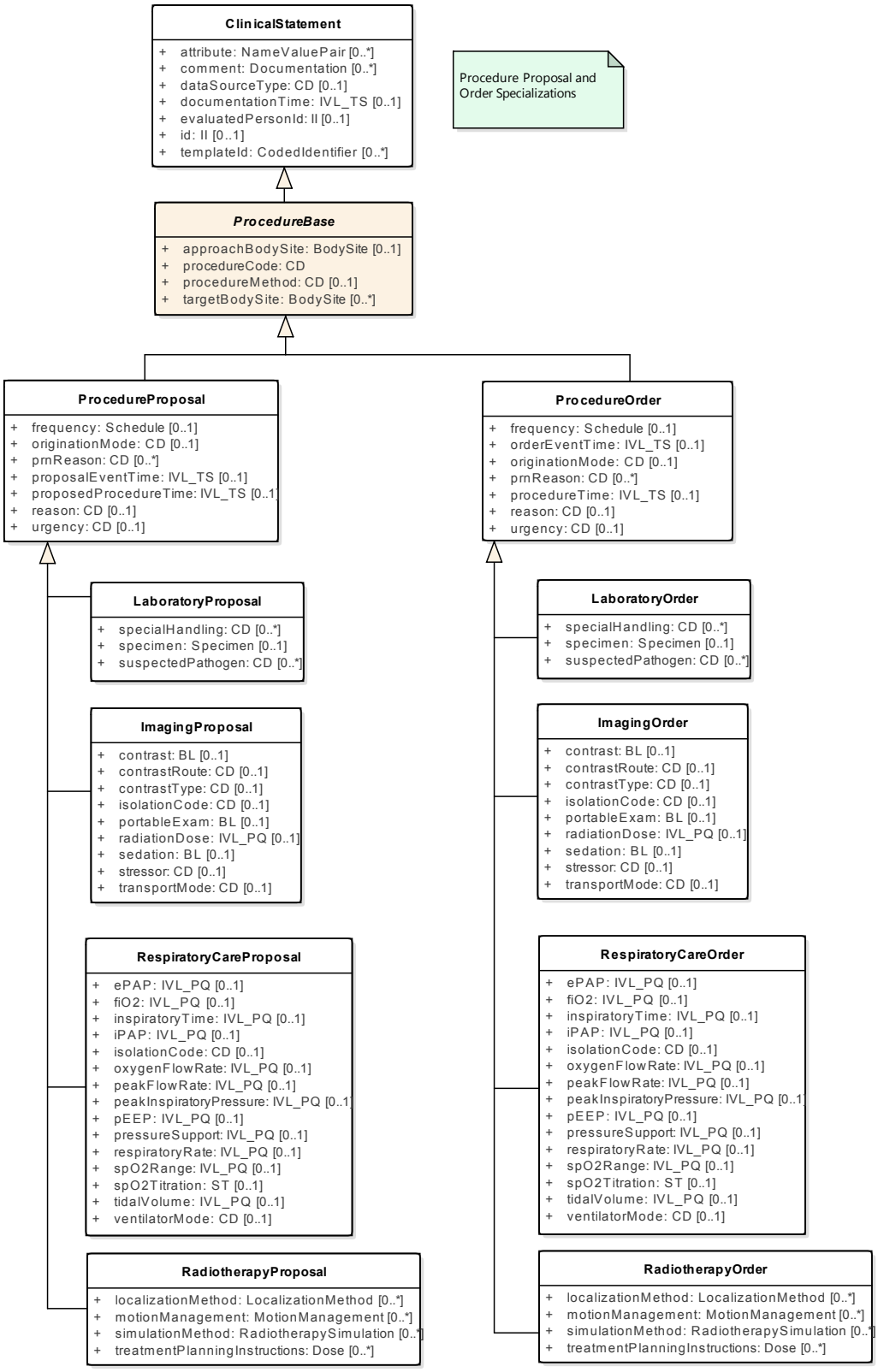

FIGURE B.13 – Représentation initiale du standard vMR (13/19)

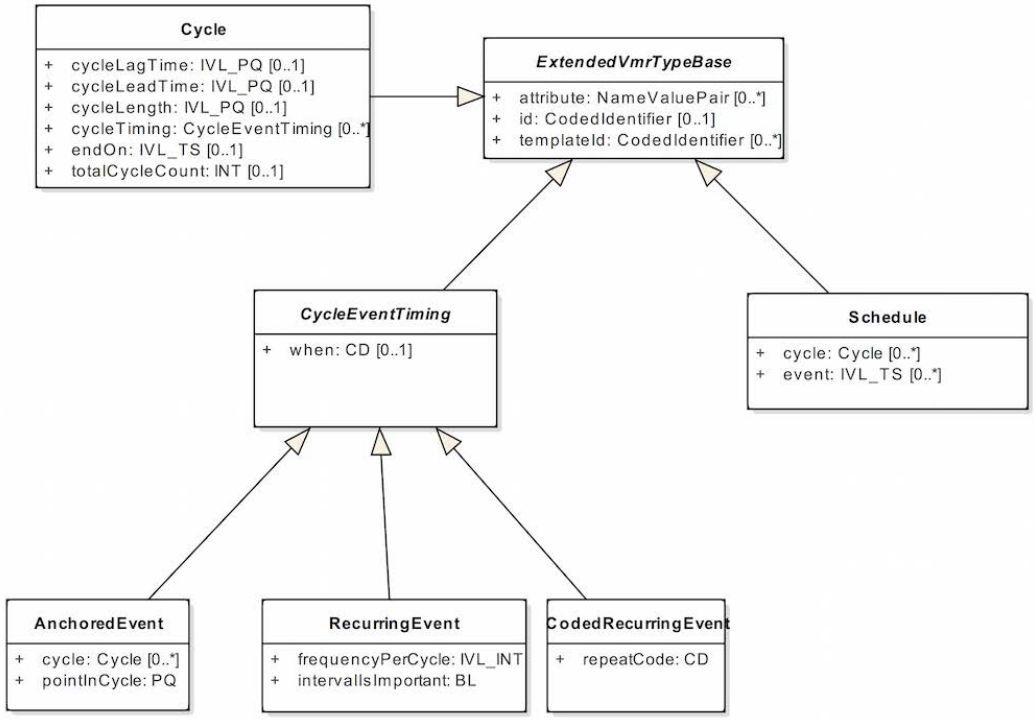

Figure B.14 – Représentation initiale du standard vMR (14/19)

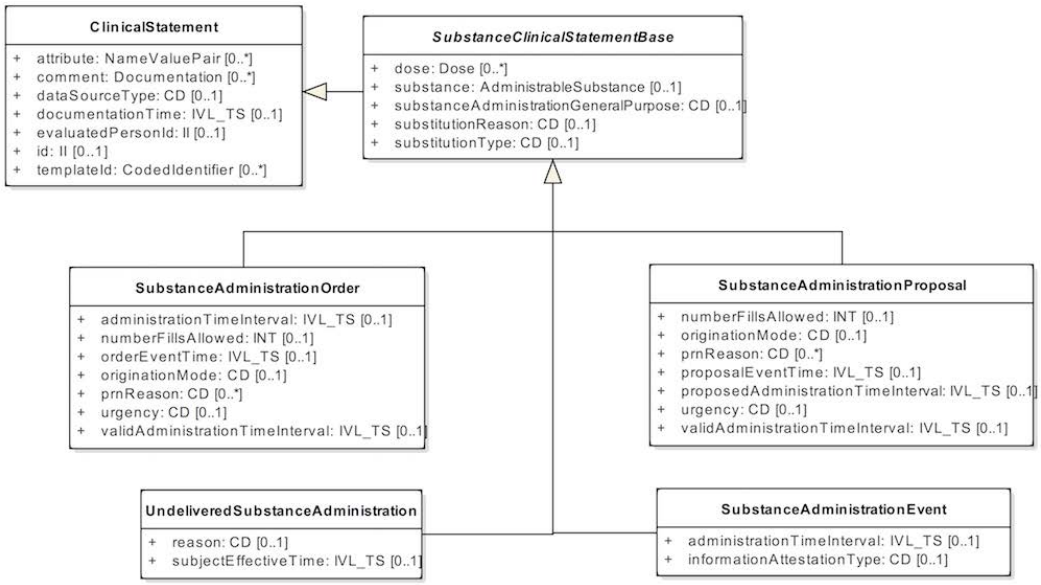

FIGURE B.15 – Représentation initiale du standard vMR (15/19)

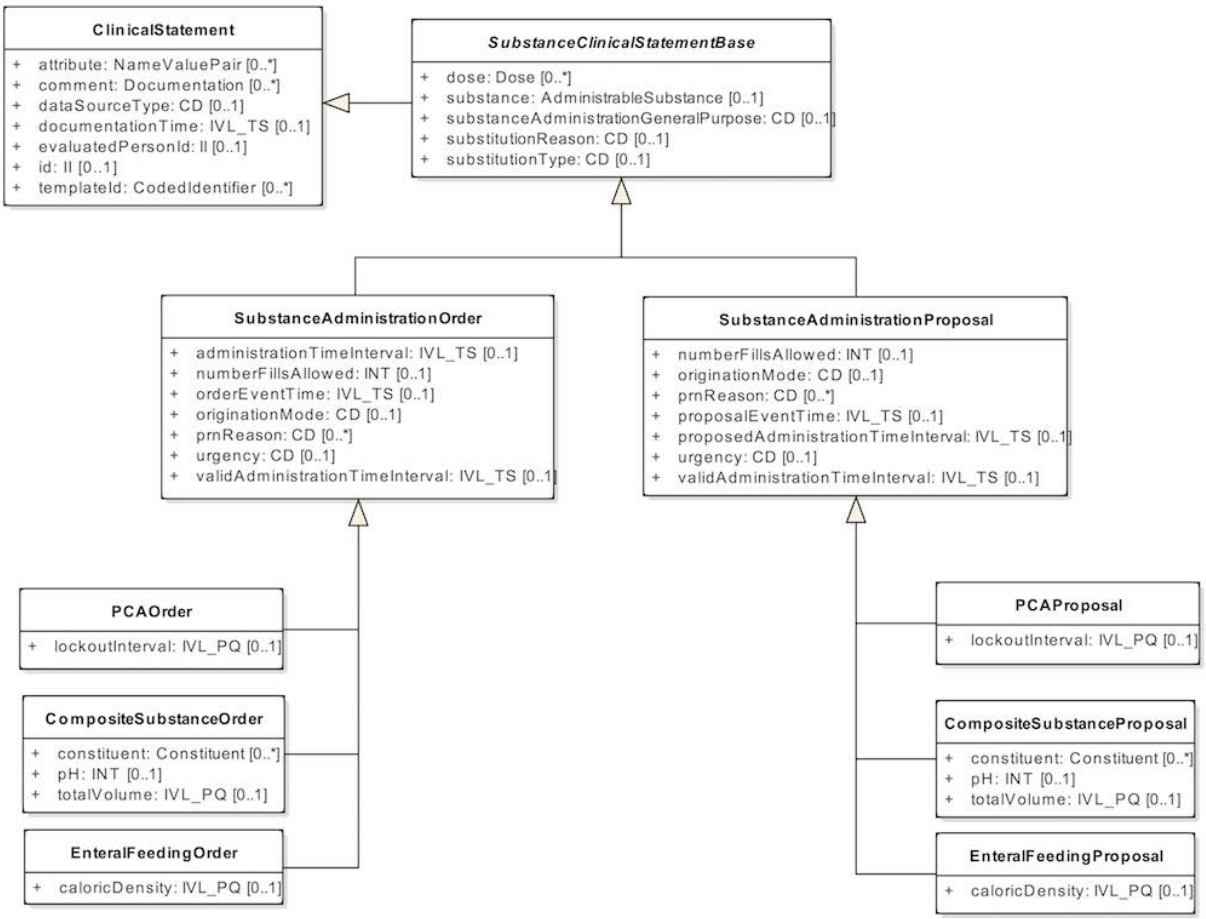

Figure B.16 – Représentation initiale du standard vMR (16/19)

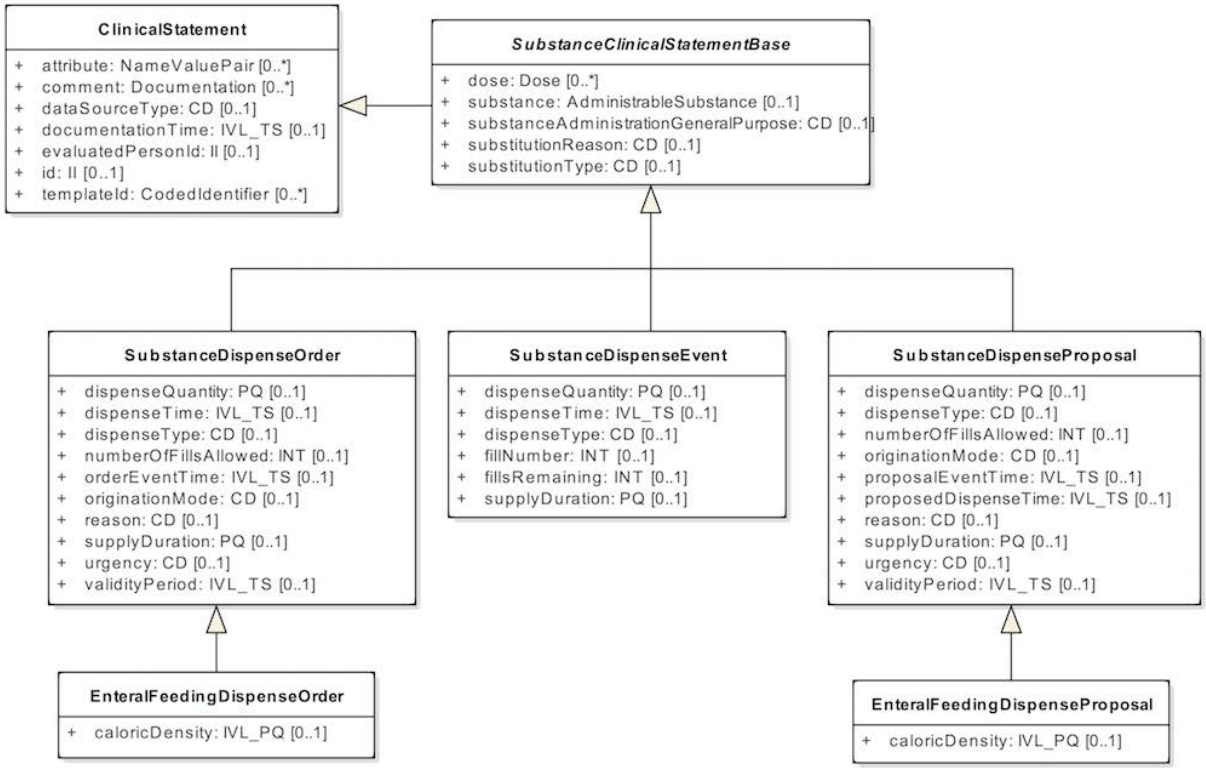

FIGURE B.17 – Représentation initiale du standard vMR (17/19)

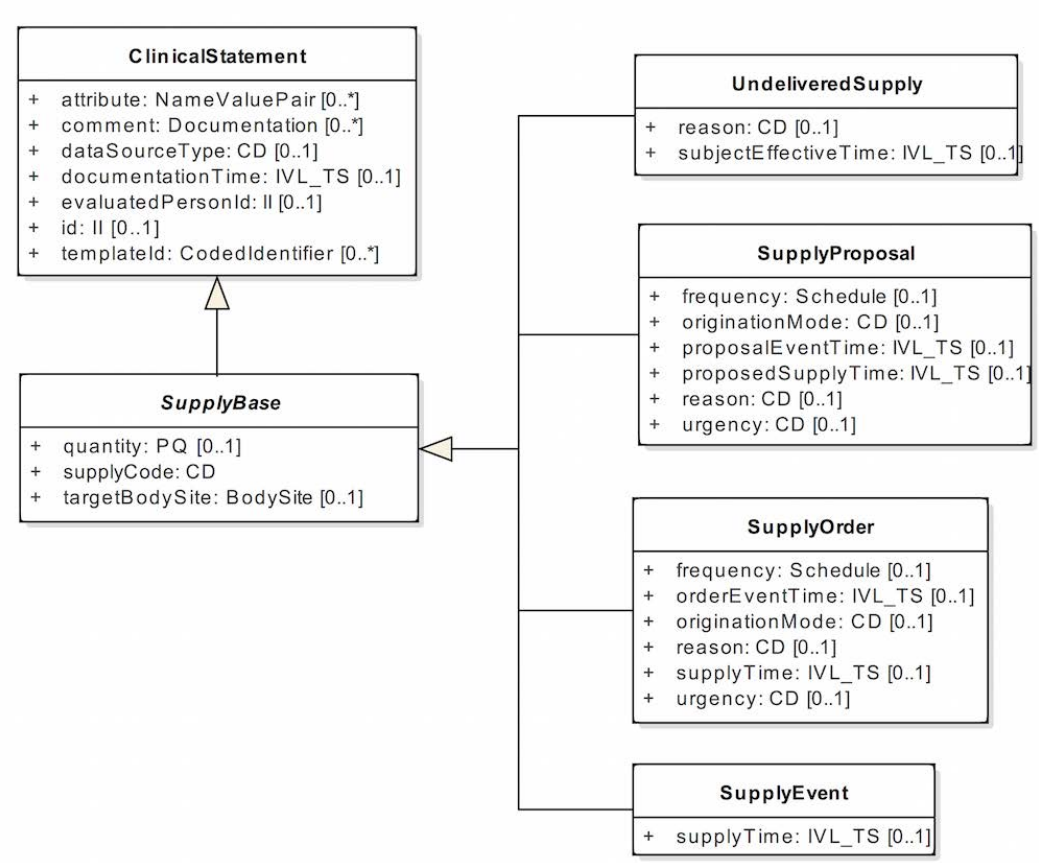

Figure B.18 – Représentation initiale du standard vMR (18/19)

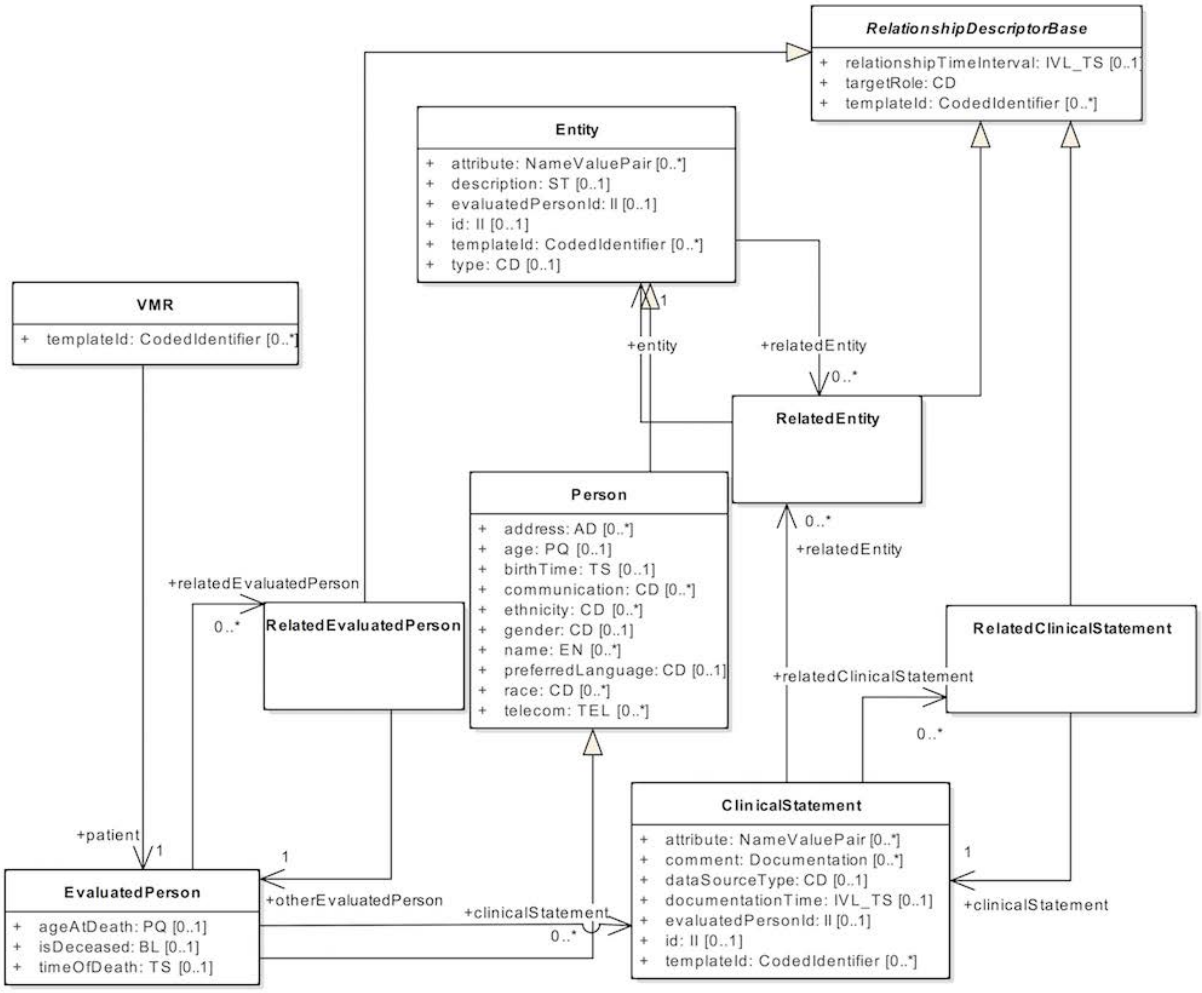

FIGURE B.19 – Représentation initiale du standard vMR (19/19)

Annexe C

Schémas Relationnels de vMR

Les schémas présentés dans cette annexe représentent les schémas vMR après l'application de la méthodologie du chapitre 5. Ils ont été générés à l'aide du logiciel MySQLWorkbench.

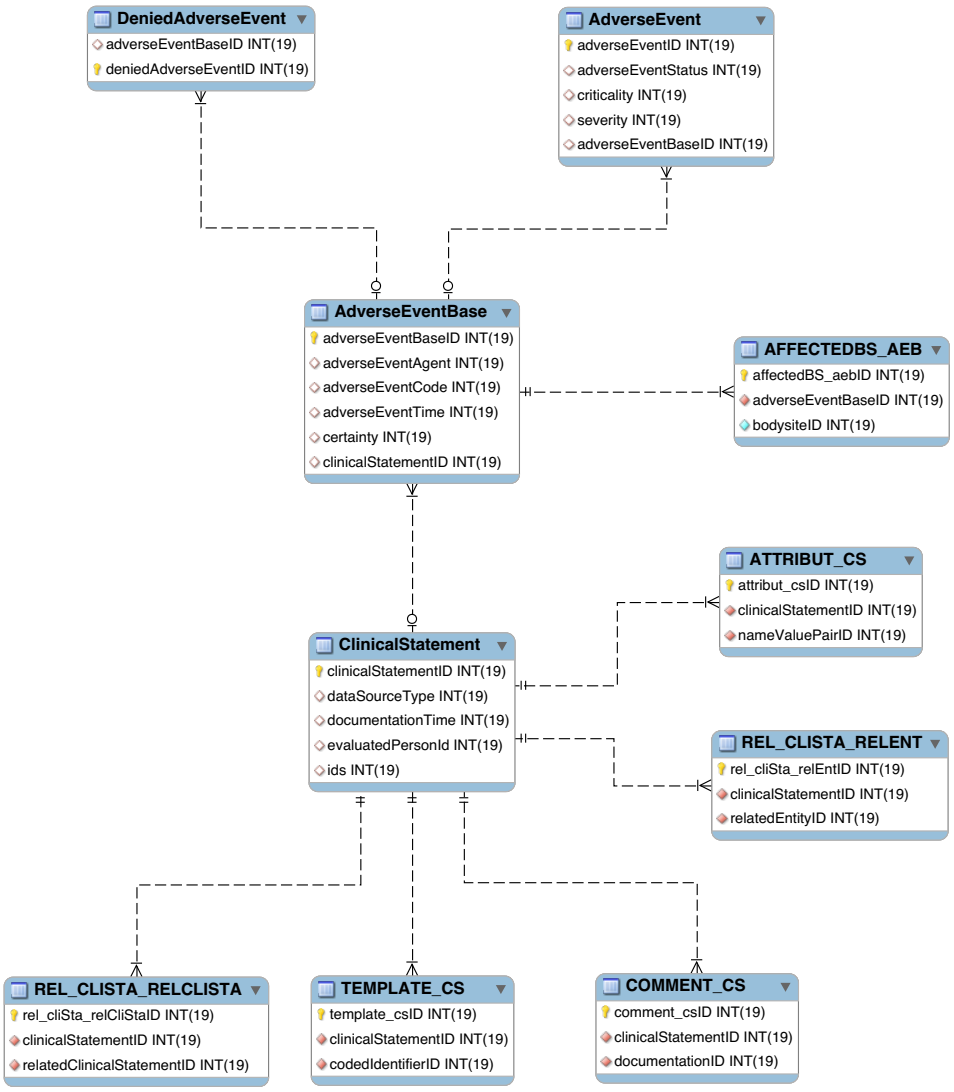

FIGURE C.1 – Schéma de vMR transformé par la méthodologie (1/20)

Figure C.2 – Schéma de vMR transformé par la méthodologie (2/20)

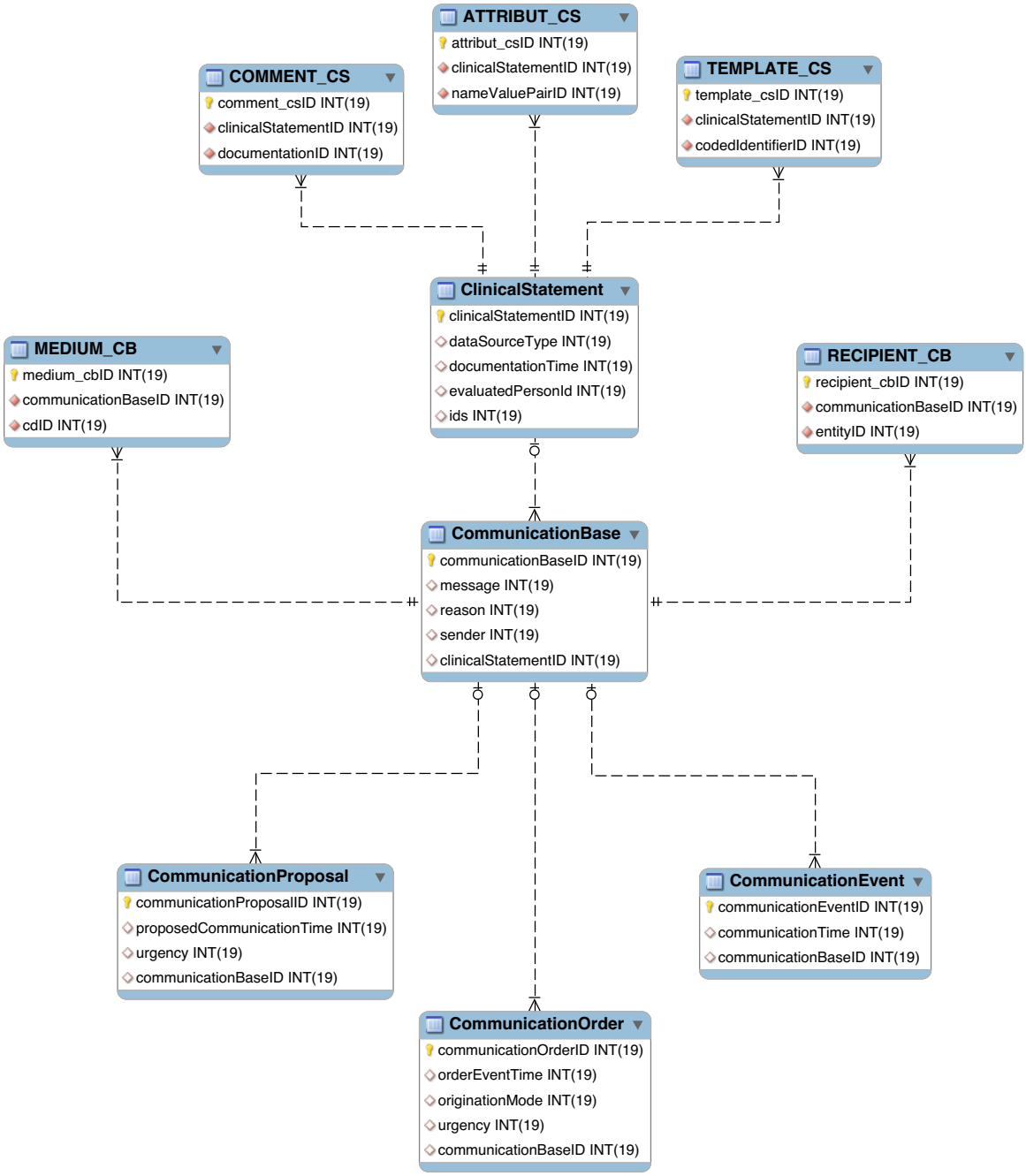

FIGURE C.3 – Schéma de vMR transformé par la méthodologie (3/20)

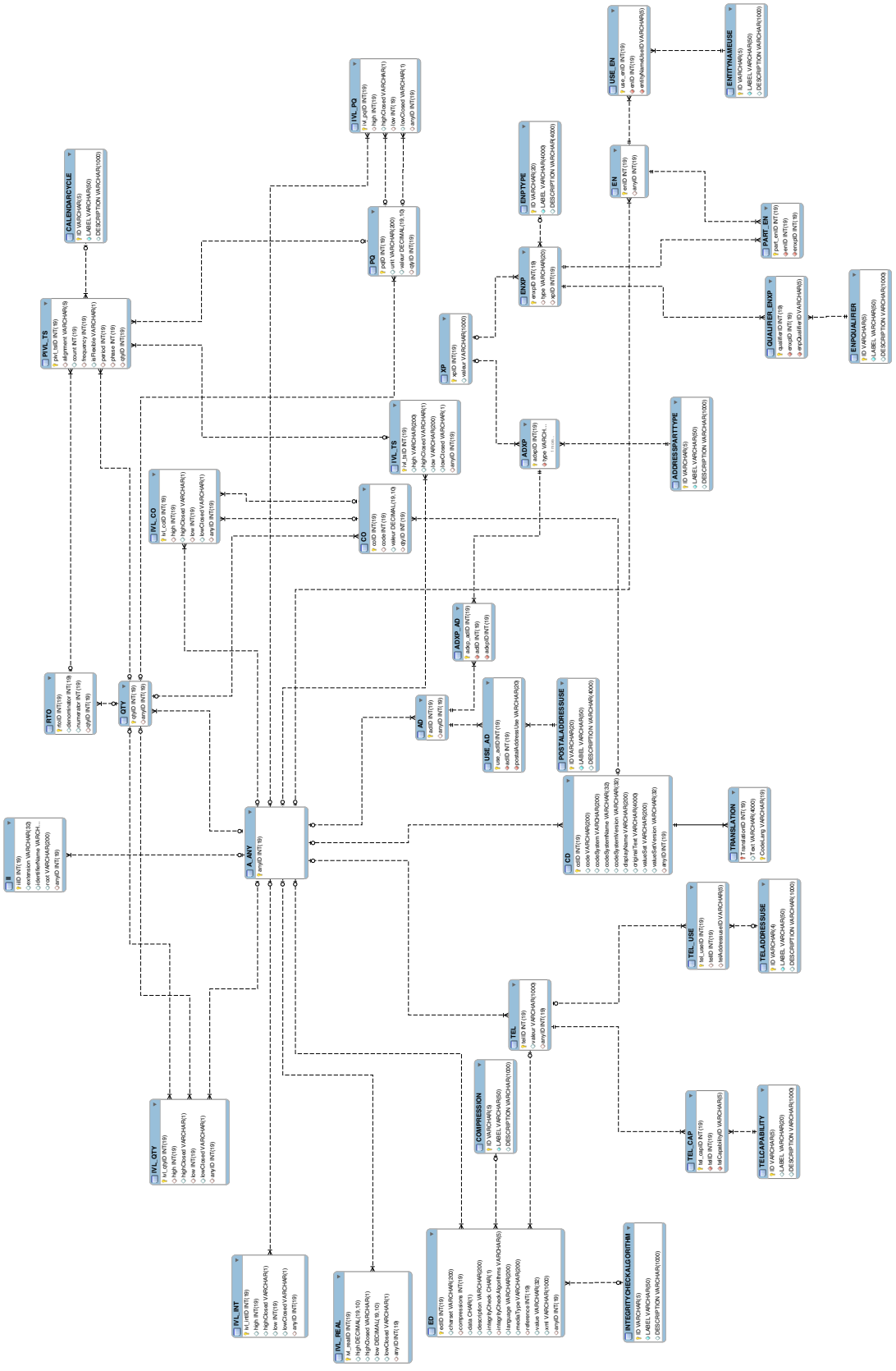

Figure C.4 – Schéma de vMR transformé par la méthodologie (4/20)

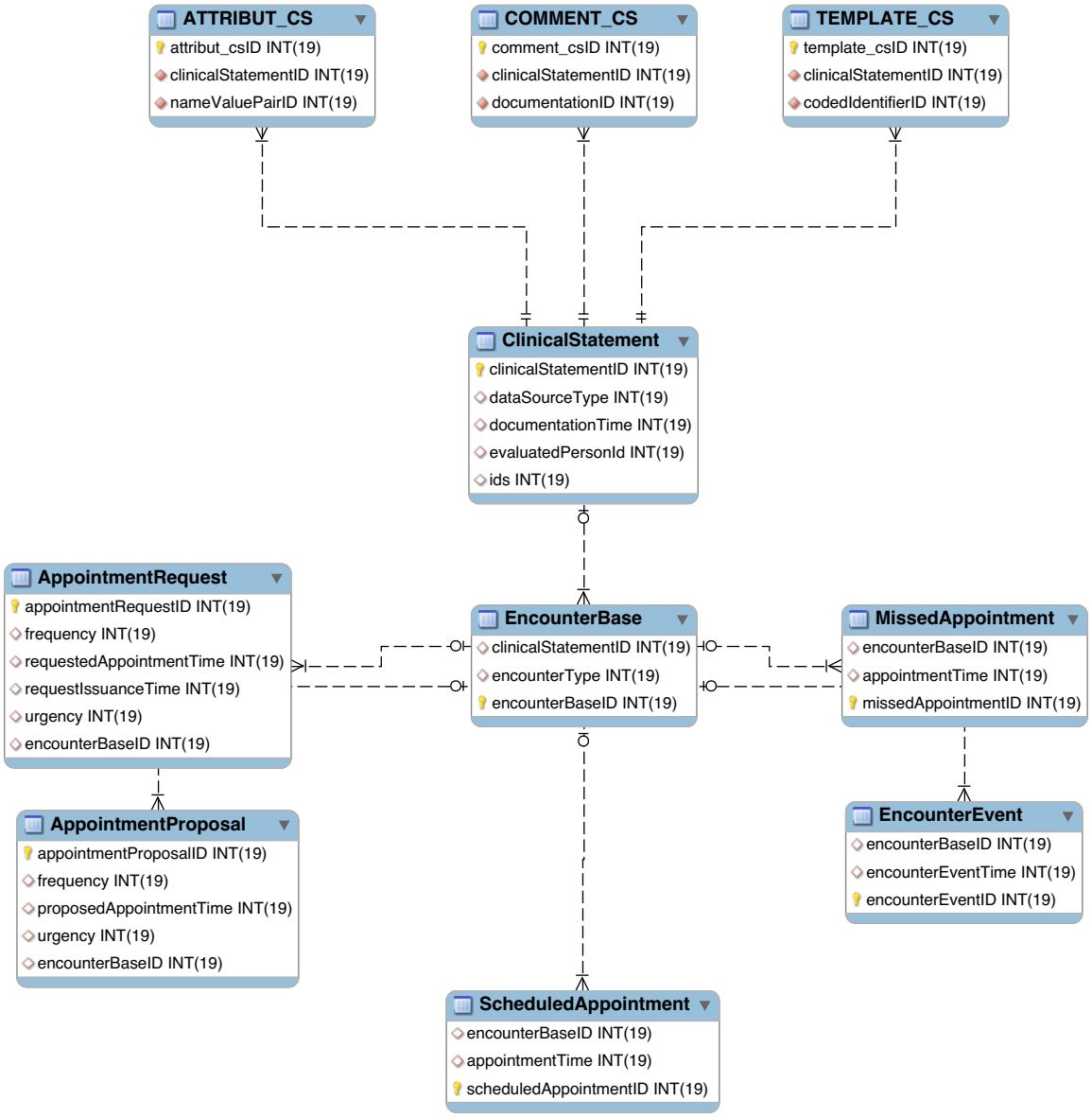

FIGURE C.5 – Schéma de vMR transformé par la méthodologie (5/20)

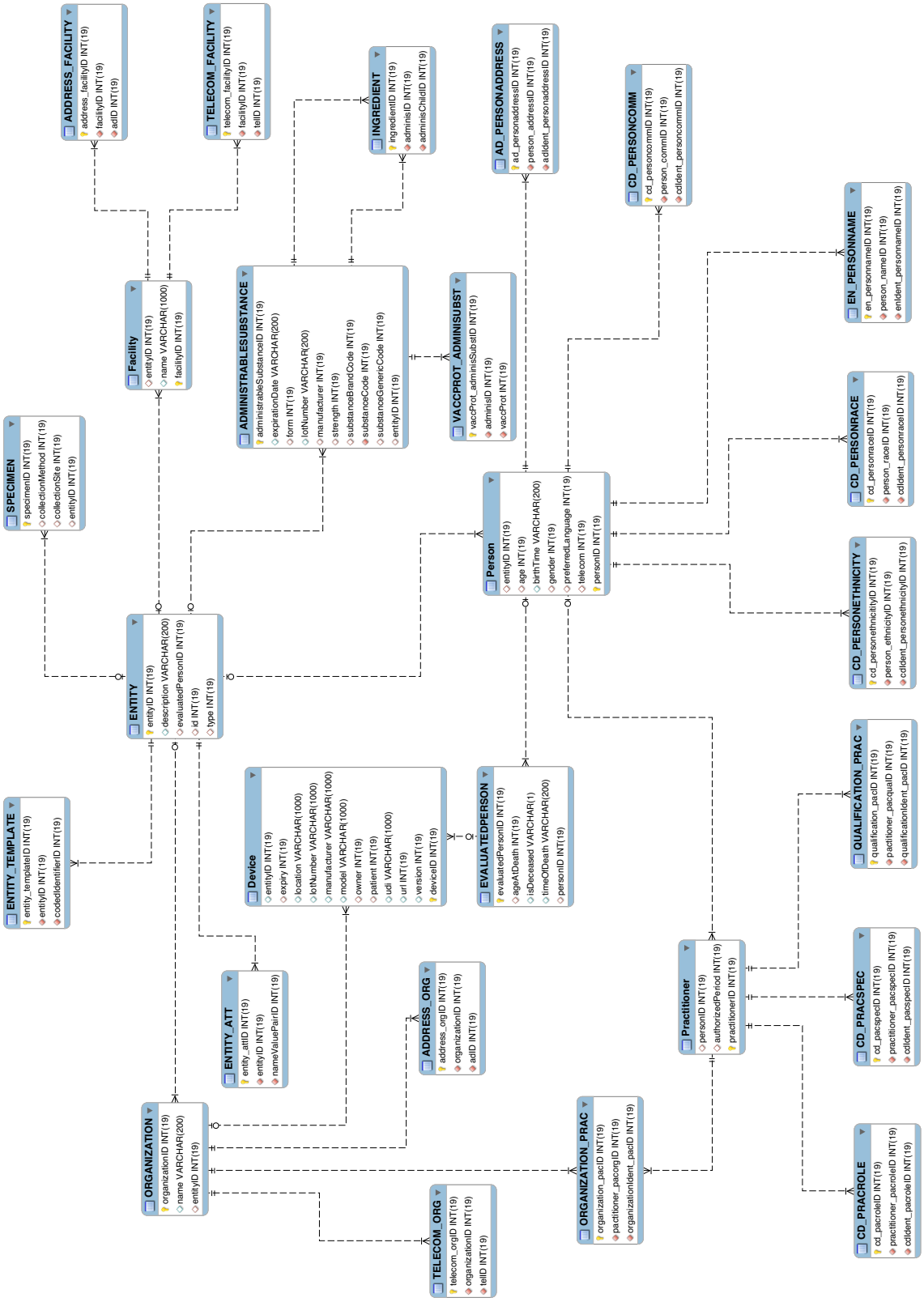

Figure C.6 – Schéma de vMR transformé par la méthodologie (6/20)

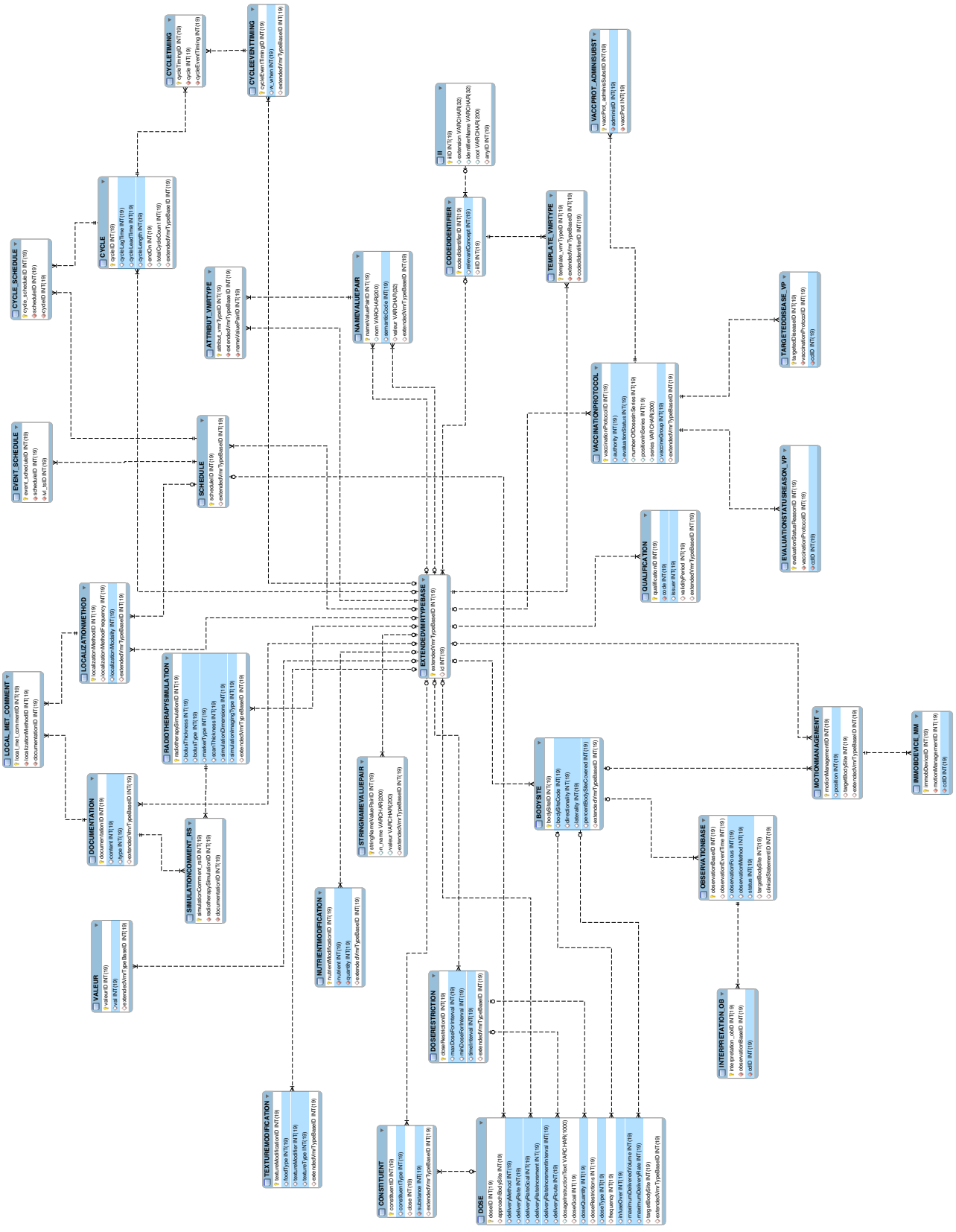

FIGURE C.7 – Schéma de vMR transformé par la méthodologie (7/20)

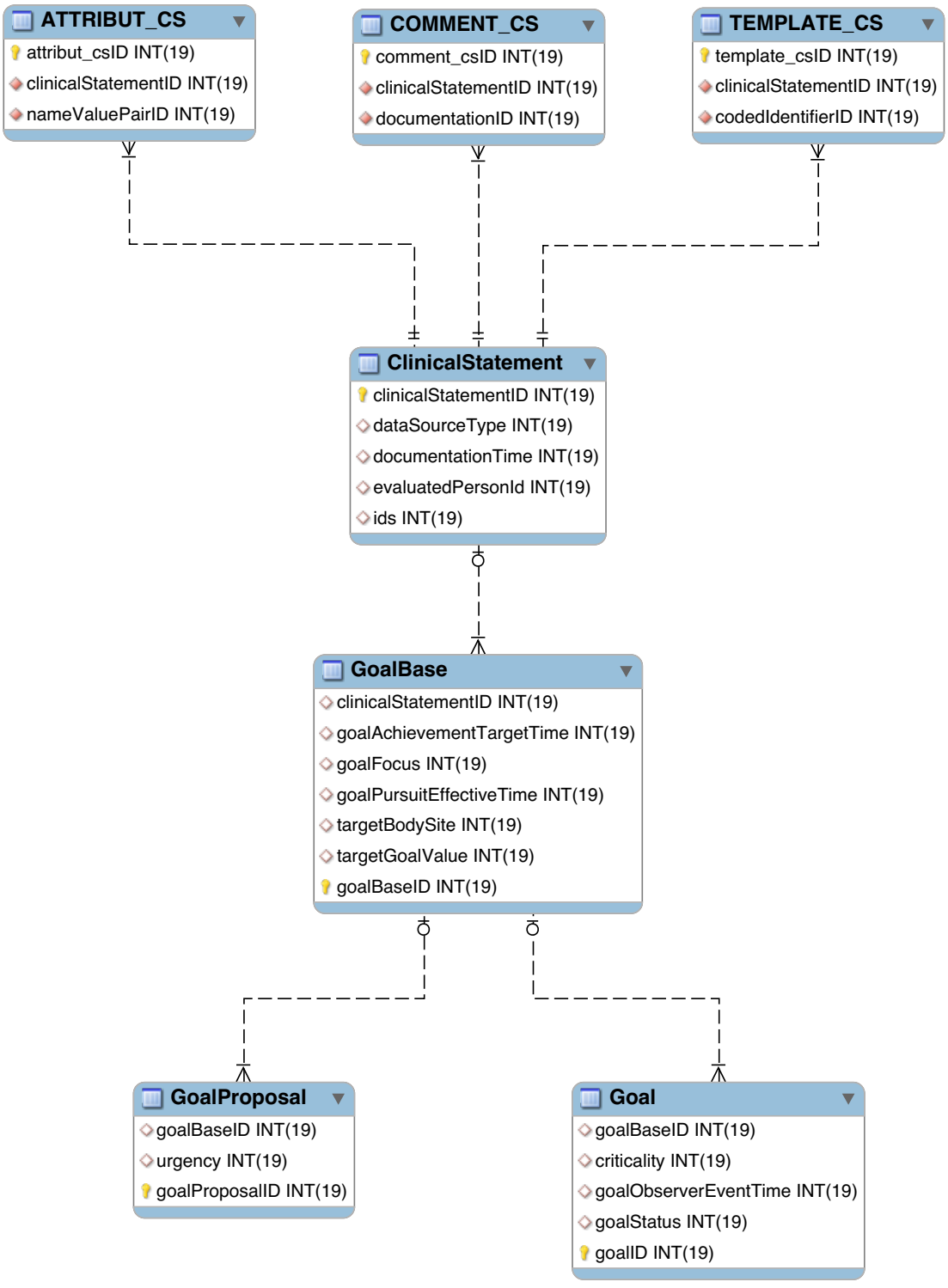

Figure C.8 – Schéma de vMR transformé par la méthodologie (8/20)

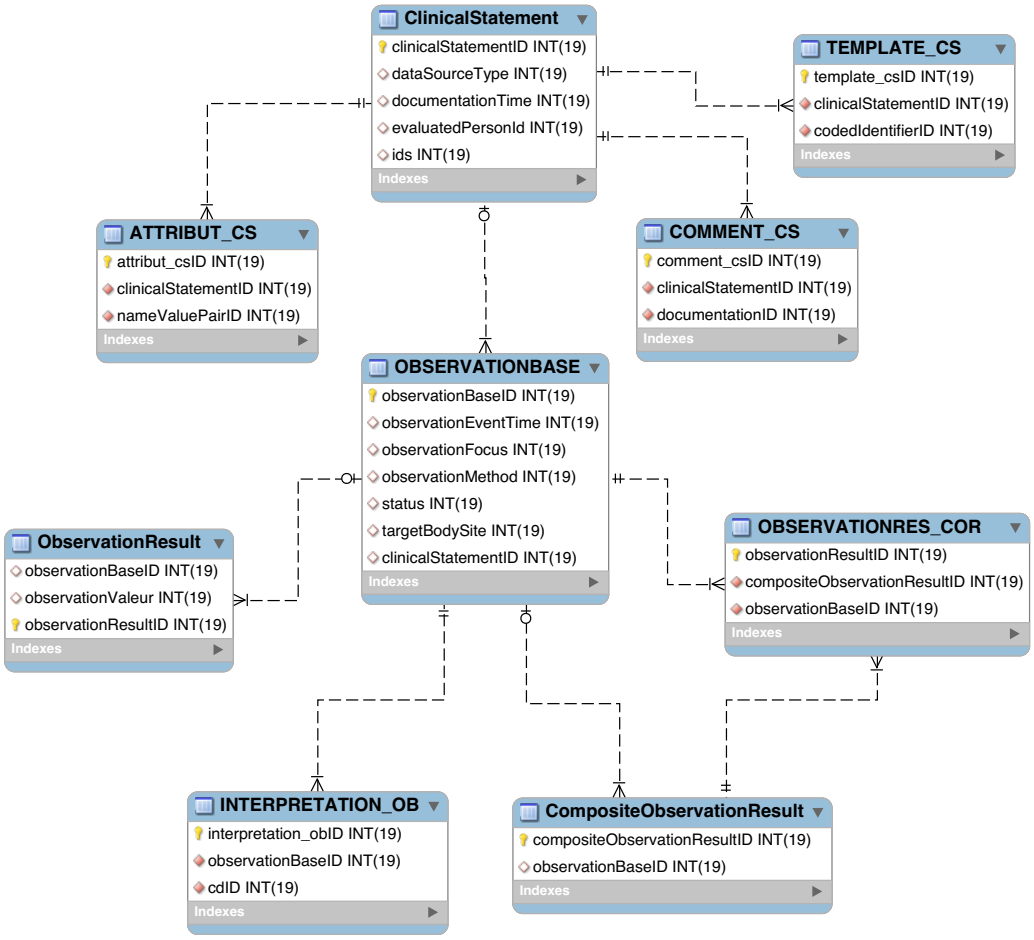

FIGURE C.9 – Schéma de vMR transformé par la méthodologie (9/20)

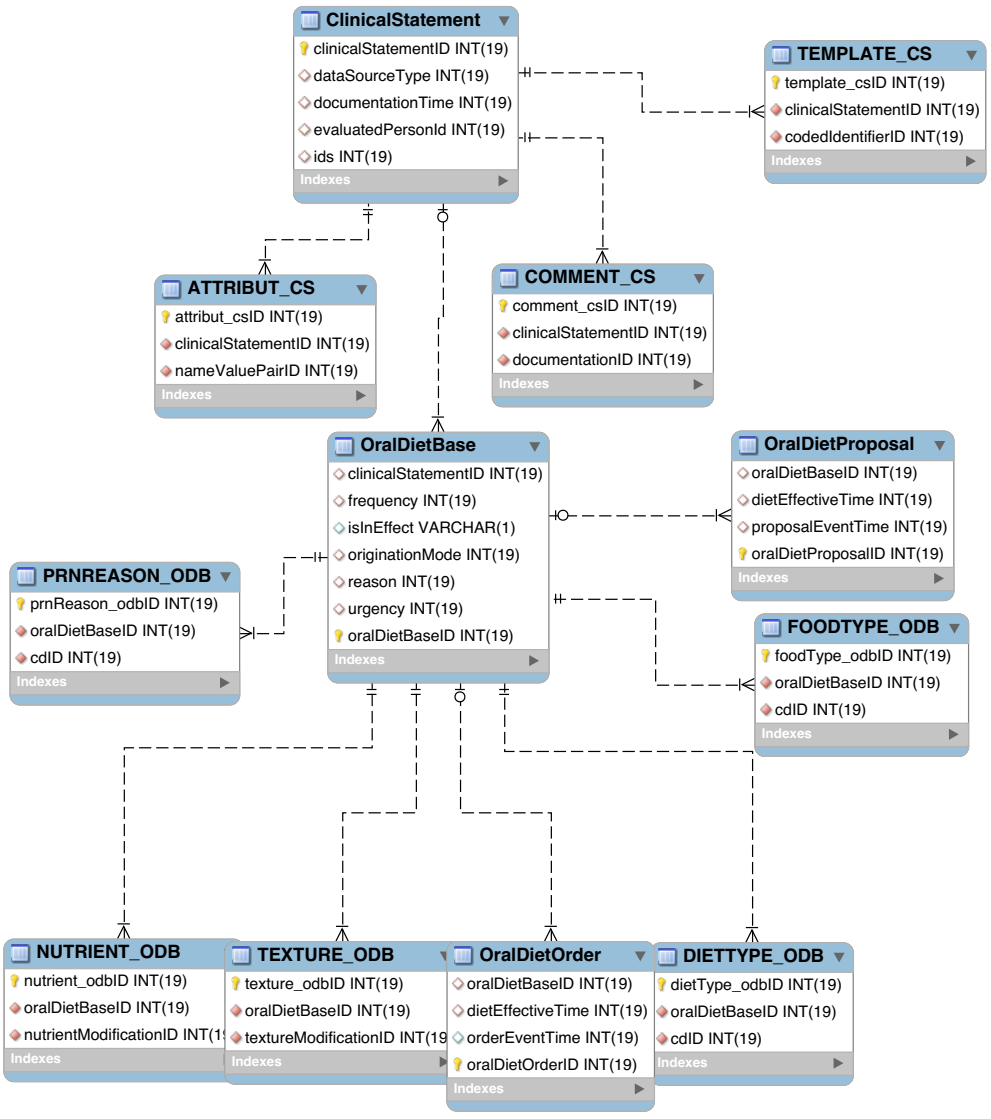

Figure C.10 – Schéma de vMR transformé par la méthodologie (10/20)

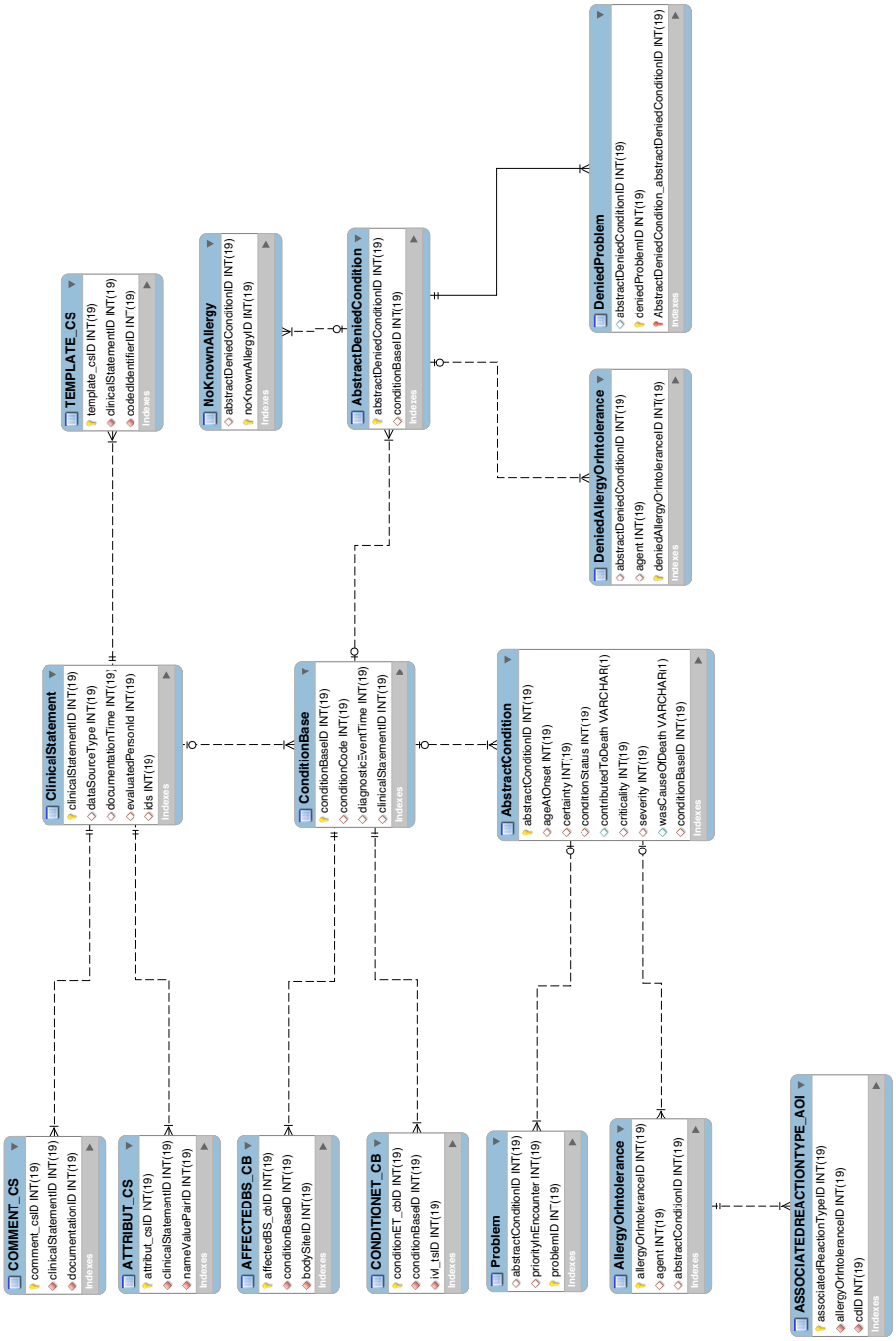

FIGURE C.11 – Schéma de vMR transformé par la méthodologie (11/20)

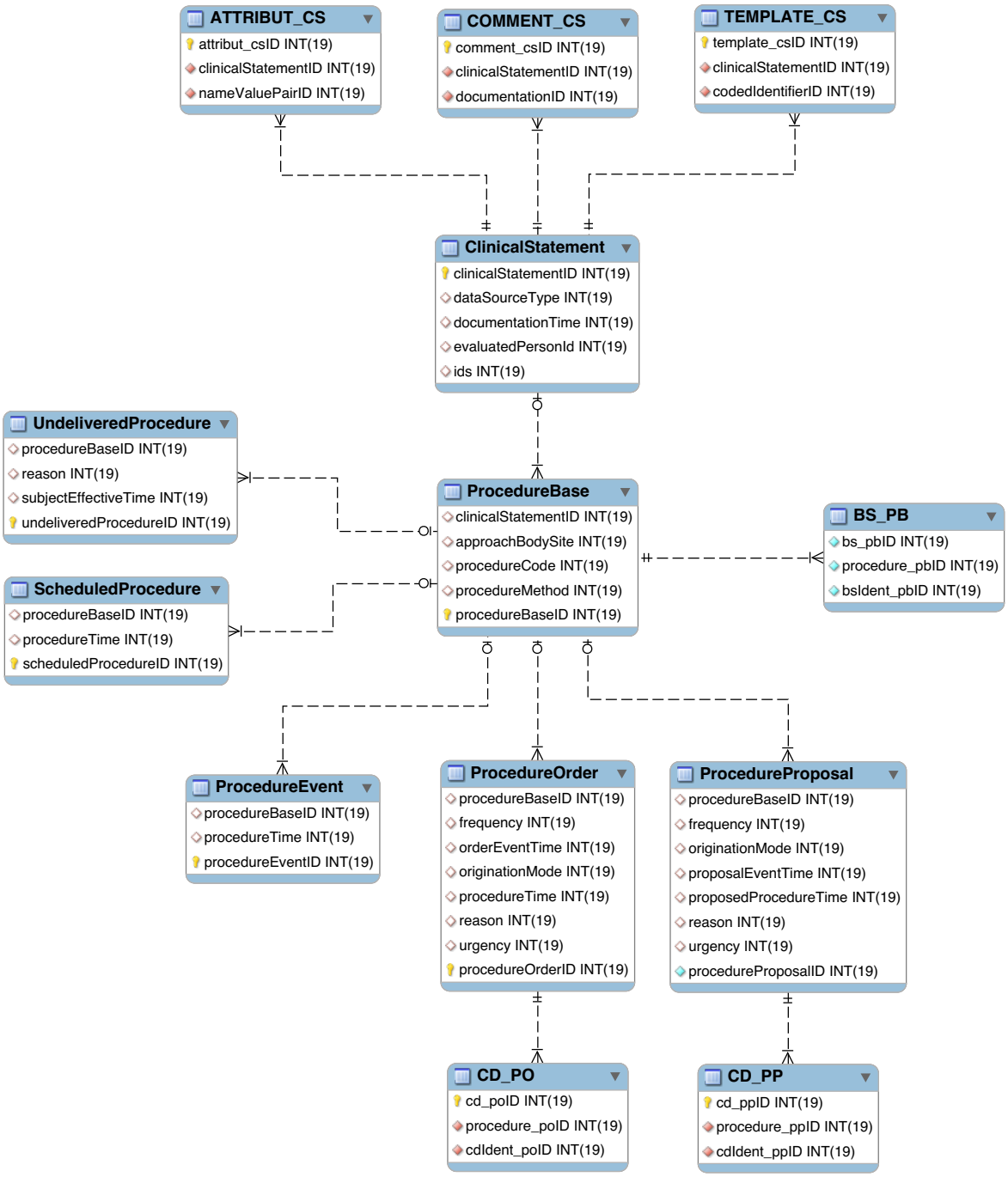

Figure C.12 – Schéma de vMR transformé par la méthodologie (12/20)

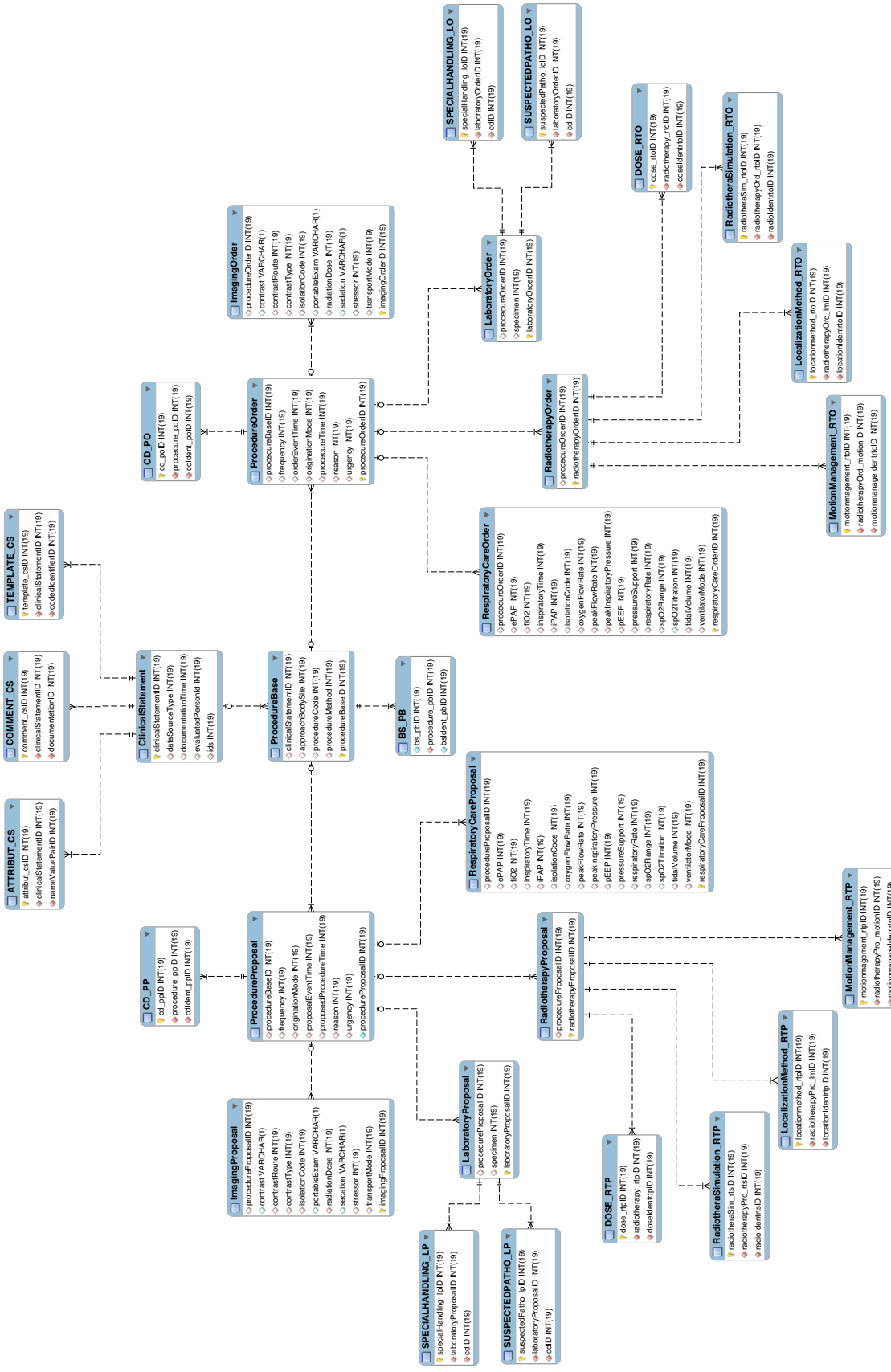

FIGURE C.13 – Schéma de vMR transformé par la méthodologie (13/20)

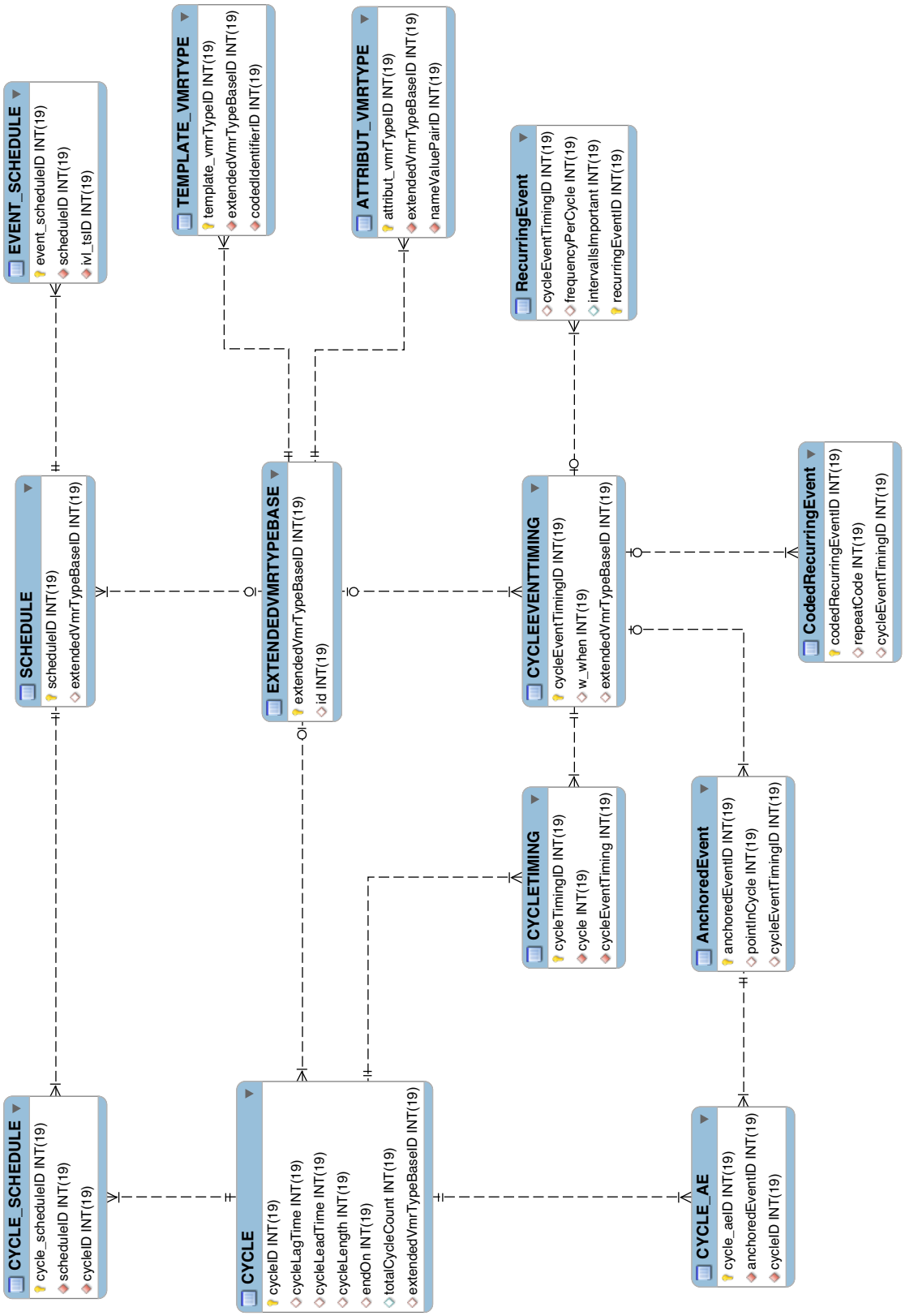

Figure C.14 – Schéma de vMR transformé par la méthodologie (14/20)

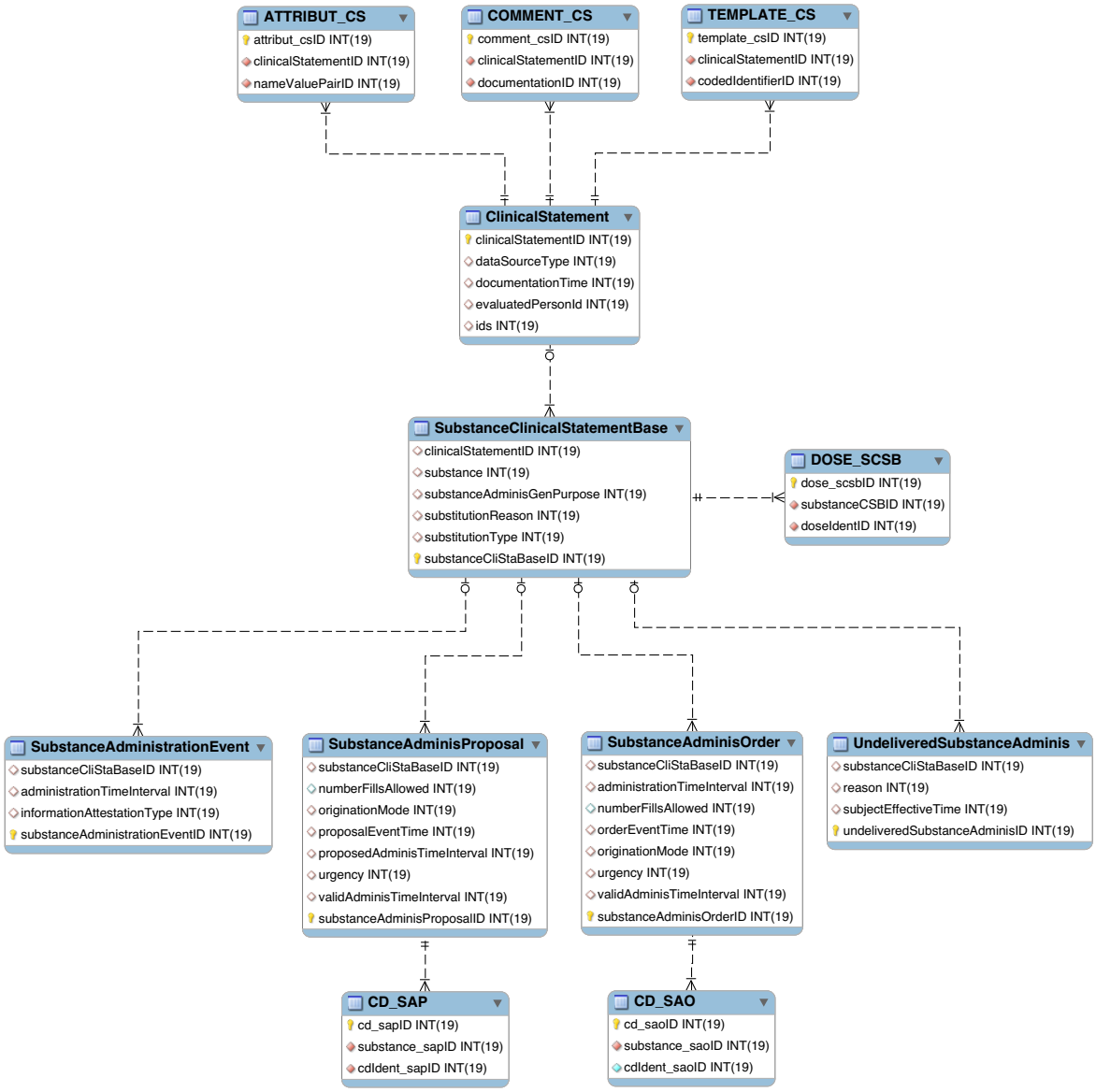

FIGURE C.15 – Schéma de vMR transformé par la méthodologie (15/20)

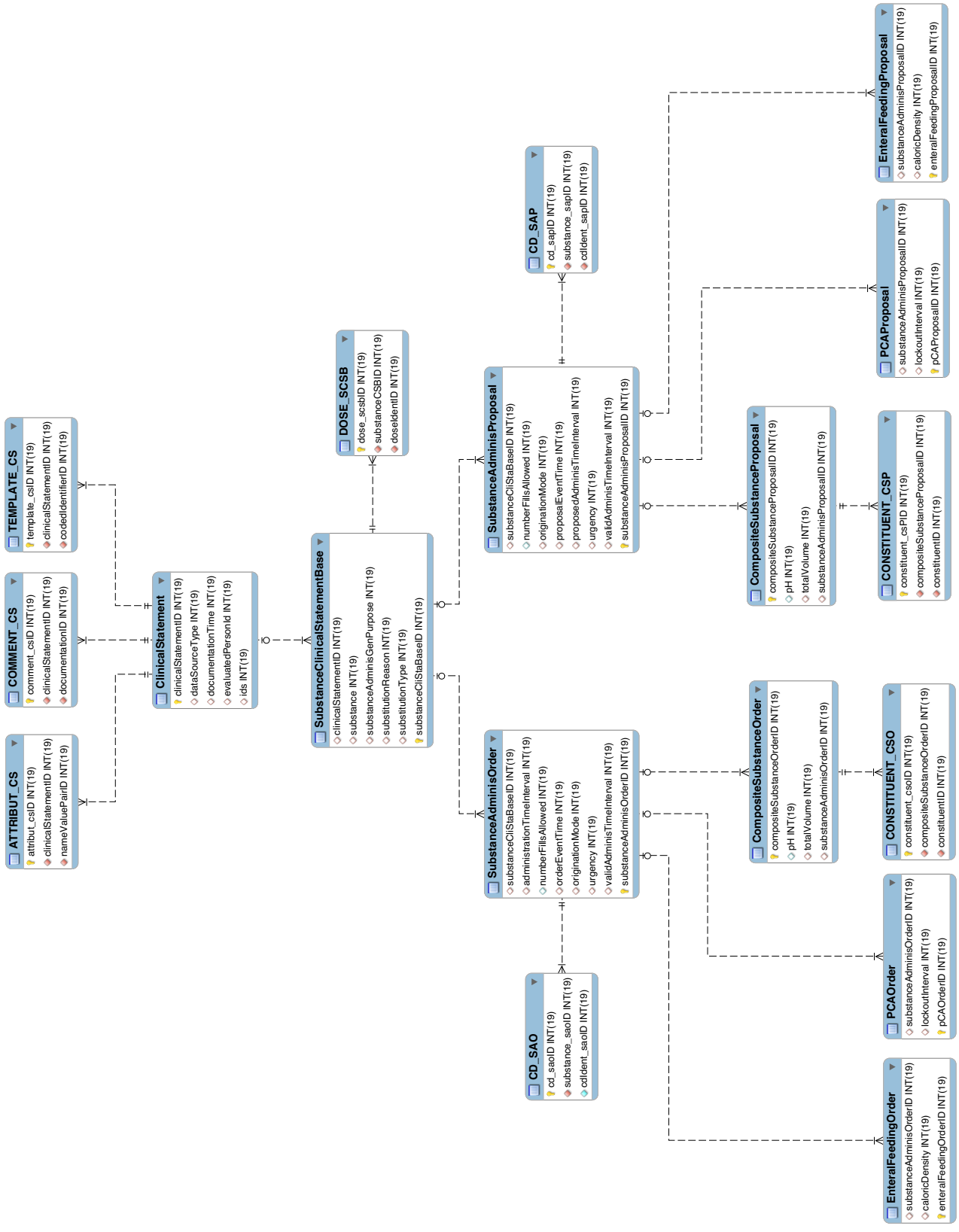

Figure C.16 – Schéma de vMR transformé par la méthodologie (16/20)

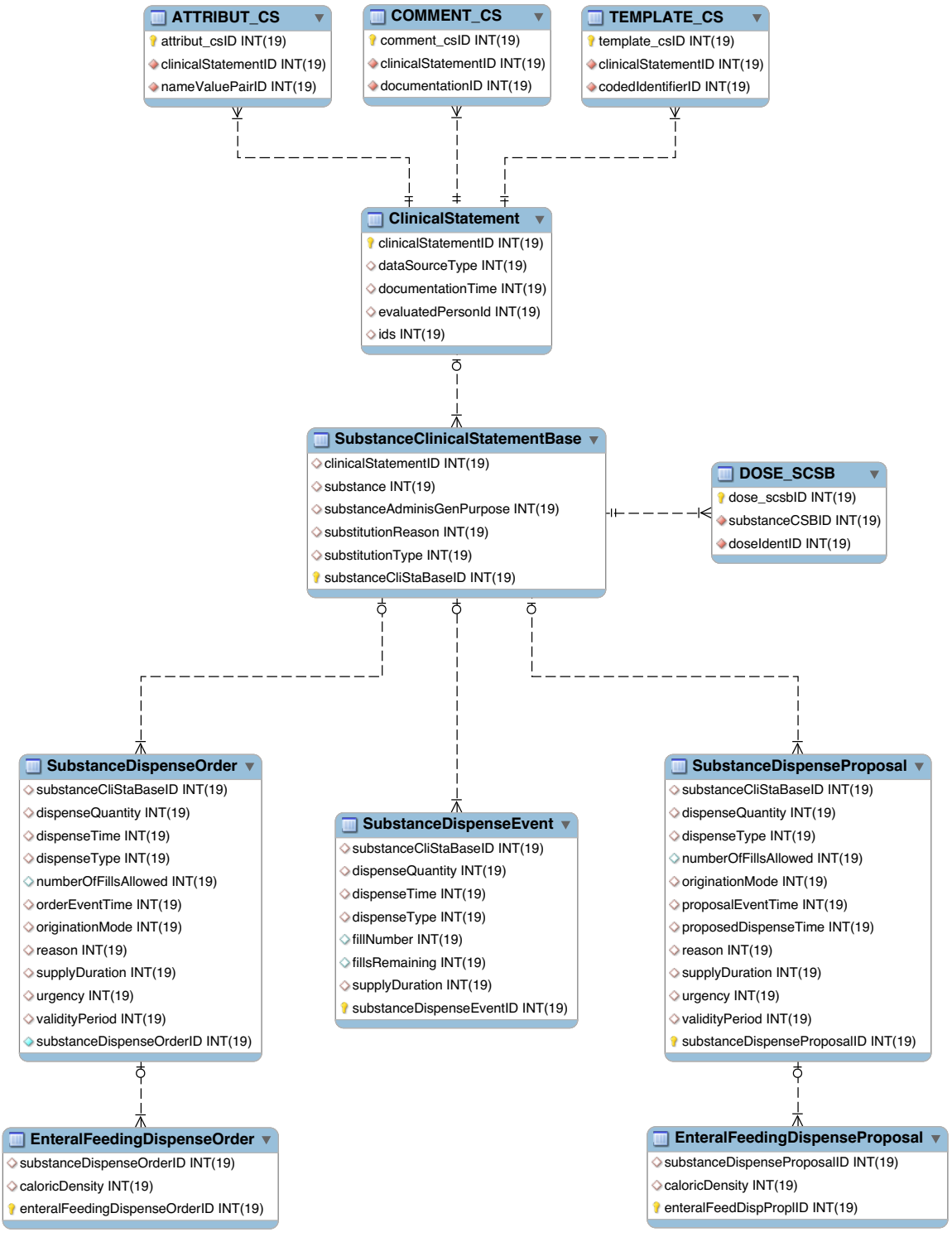

FIGURE C.17 – Schéma de vMR transformé par la méthodologie (17/20)

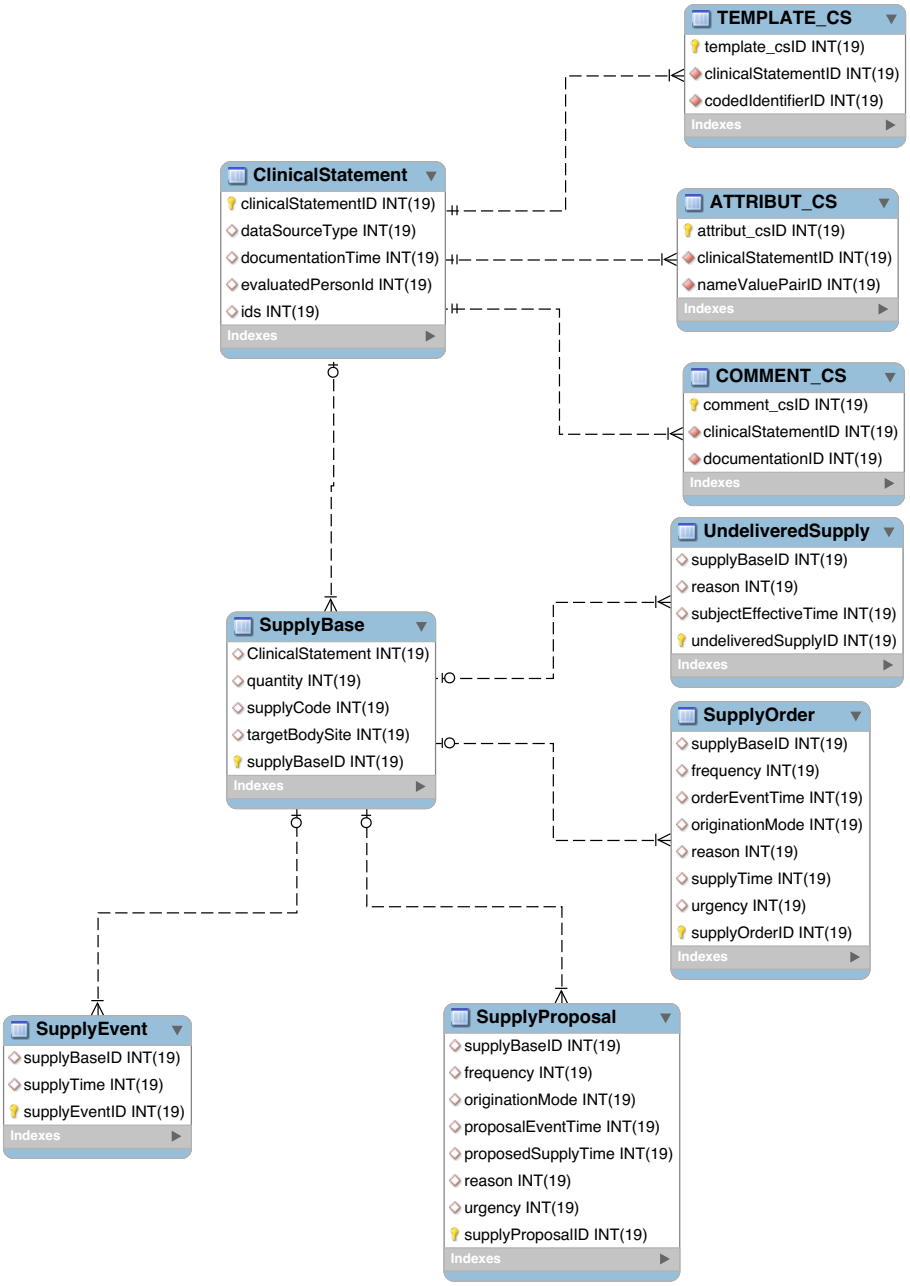

Figure C.18 – Schéma de vMR transformé par la méthodologie (18/20)

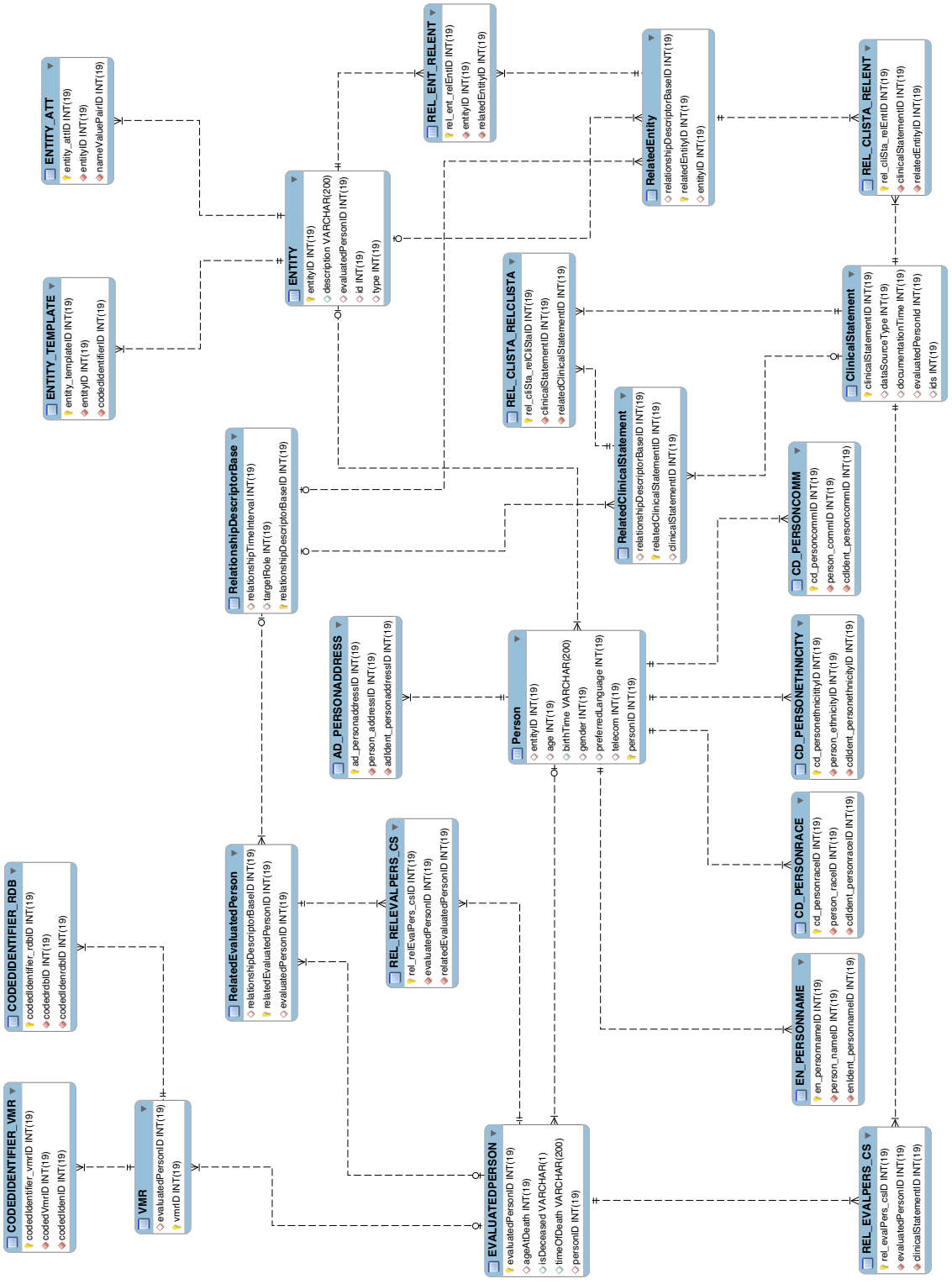

FIGURE C.19 – Schéma de vMR transformé par la méthodologie (19/20)

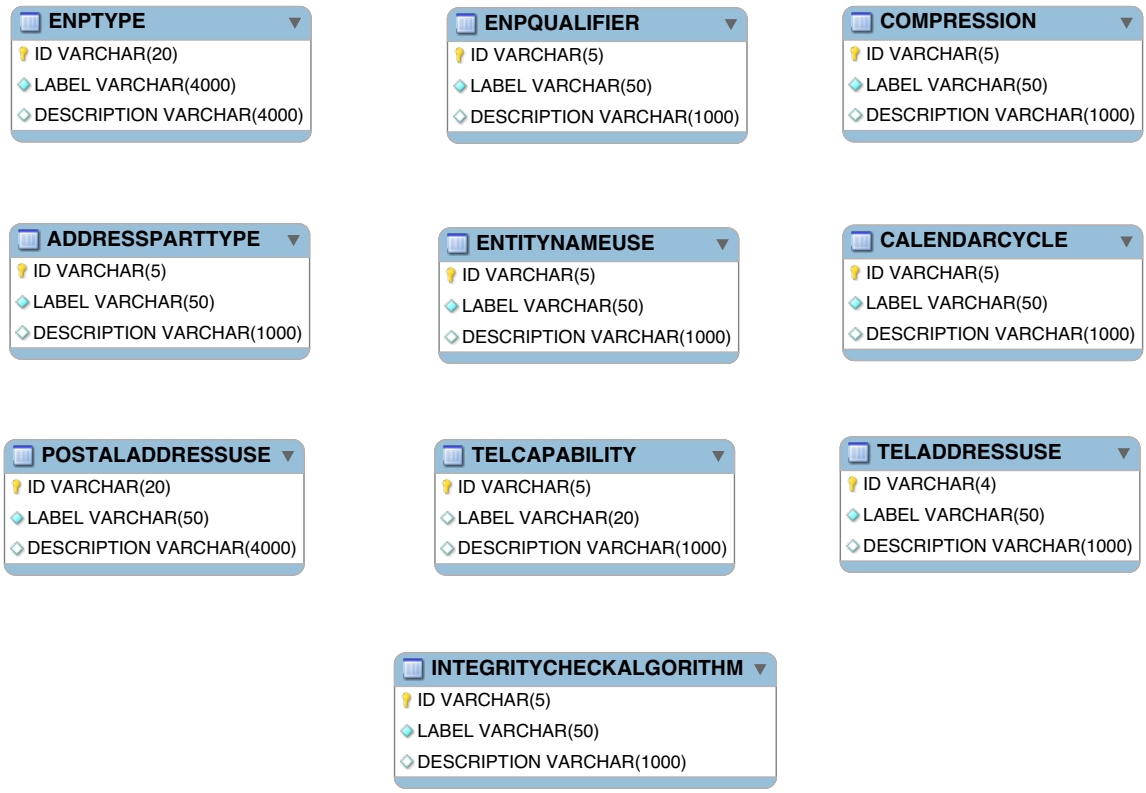

Figure C.20 – Schéma de vMR transformé par la méthodologie (20/20)

Annexe D

Contenu du CD-ROM

Cette annexe présente le contenu du disque fourni avec le mémoire :

1. Le mémoire en version .pdf ;
2. Les schémas originaux et la documentation de vMR ;
3. Les schémas transformés de vMR (après l'application de la méthodologie) ;
4. Le code SQL final pour Oracle ;
5. Le code source de l'application RetroStudy ;